MASTER THESIS

TITLE: Application of Machine Learning for energy efficiency in mobile networks

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: David Sesto Castilla

ADVISOR: Eduard Garcia Villegas

DATE: September, 14th 2017

Title: Application of Machine Learning for energy efficiency in mobile networks

Author: David Sesto Castilla

Advisor: Eduard Garcia Villegas

Date: September 14th 2017

Overview

Future generation networks (5G) will bring a new paradigm to network management, as the networks themselves will suffer evident changes that will imply new requirements in upper layers.

The 5G-XHaul project, framed under the Horizon 2020 European research and innovation programme is focused on providing dynamically reconfigurable optical-wireless backhaul and fronthaul architectures with a cognitive control plane for small cells and cloud-RANs. One of the objectives contained under that premise consists in the design of new network management strategies for mobile networks, subject to which this thesis contributes.

Making use of new technologies and techniques, we can deploy a multi-tier network with a lower layer of small cell deployments that are managed through a dynamic system that can automatically perform certain operations over that network. Machine Learning is an increasing trend in this field, can help with the process by making use of the data collected from the network, obtain useful knowledge, and create predictive models that can tell us the state of the network in the near future.

For the development of this project, we have collaborated with COSMOTE, one of the main telecommunications companies in Greece, who have provided us with several data sets of a real network deployment in the centre of Athens. With these data, several predictive models have been created to predict the state of the network during certain time intervals and act in consequence. Many different applications can be found for those algorithms, although one of those that is a hot topic nowadays is energy efficiency. To work on that field, the prediction models where used to create a dynamic system that turns cells on and off dynamically, depending on the expected traffic, in order to achieve notable energy savings.

Finally, a simulation environment was developed, based on the real traces from the COSMOTE network, in order to test the proposed network management techniques in a large number of different scenarios. This simulator generates realistic random scenarios from which several statistics can be extracted, with the aim of measuring the performance of the algorithms developed during the earlier stages of the project.

Working with different tools and environments, this project studies the best data analysis and Machine Learning techniques regarding network usage data. From that data, prediction models are created, which can be used for many different and interesting applications. The one chosen for this thesis is the design of an energy efficient management system for dense small cell deployments. Finally, results are collected, and the validity of the proposed strategies is proved.

Acknowledgement

The original idea of this project has to be granted to Eduard Garcia (director of my thesis at EETAC). Eduard got me in touch with the research group at I2CAT with which we have collaborated during the development, and he has guided me through the project. He has always been involved in the thesis, offering advice and guidelines when required.

I would also like to mention Daniel Camps and Ilker Demirkol, the members of the I2CAT team involved in the 5G-XHaul project, who have always been available to help me with the early steps of the development.

Finally, I would also like to thank my family and friends for supporting me throughout the process, and Sandra, for being an essential inspiration always and forever.

Thank you all for the opportunity and the support.

INDEX

FIGURES INDEX

CHAPTER 1. INTRODUCTION

1.1. The project

Since the first generations of cellular networks, vendors collect enormous amounts and varieties of data, which are stored in various formats, and later processed or ignored, just to be erased after short periods of time. All these data, which until some years ago were neglected, can provide useful information about how a network works, which are its strengths and weaknesses, etc. In this thesis we argue that these data can even contribute to an enhanced real-time management.

The data collected and stored by the operators belong to different categories (network, user, content, external data), are collected by different entities (network nodes, UE reports) and provide information about network traffic, resource access, QoS, cell availability, etc. The usage of these data may vary depending on the functionality we have in mind, considering both scenarios, where long-term information is required (such as network planning actions based on the records of the network over time) or real-time optimization techniques based on the latest metrics. Raw data are useless and difficult to work with, reason why there are some initial steps that have to be performed over the sets of information before starting to work with them.

The paradigm that the approach of the fifth generation networks (5G) is bringing requires the introduction of new smart mechanisms capable of analysing and correlating multiple data sources in order to extract relevant information. The scope of this project, then, consists in studying the traces of several datasets from the COSMOTE (one of the main telecommunications companies in Greece) network in Athens, build appropriate prediction models based on that data, and test some of the possible applications of such systems, for example an energy-saving mechanism consisting in dynamically switching off those cells that are unnecessary at a given time.

In order to work on that field, we used tools specialized in the field of data mining and machine learning, where the data sets provided by COSMOTE in the frame of the European project 5G-XHaul could be studied. Moreover, a simulator software tool was developed in order to test the models extracted from the data analysis, and apply them into the field of energy-saving techniques.

As the main aim of this project, we wanted to study the data extracted from a real network, see their potential, and consider a proof case of an application of such study in a realistic environment. In order to achieve these general goals, the following objectives were defined:

- Study of the literature regarding the application of Machine Learning techniques in network management.
- Familiarization with Data Analysis and Machine Learning tools, as long as with the main techniques, algorithms and working methodologies.
- Familiarization with the real network of COSMOTE in centre Athens, through several datasets.
- Generation of models based on the traces that can predict the network behaviour.
- Application of the previously mentioned models on a specific proof of concept: an energy saving technique consisting in switching off cells that are predicted to have a low load.
- Programming a simulation environment adapted to the requirements of the project and focused on the study of the effect caused by switching off cells dynamically.

With those objectives in mind, the project was planned into two main parts. The first of them consisted in the study of the data sets and use of Machine Learning techniques to obtain appropriate models in the scope of the project. The second step consists in making use of those models in one of the many possible research areas, in our case, energy efficiency in network management. To do so, a simulation environment capable of producing realistic scenarios was prepared, were different algorithms could be tested and the final results were obtained and discussed.

1.2. 5G-XHaul

This thesis is framed within the 5G-XHaul project [1], a project that has received funding from the European Union's *Horizon 2020* program [2], oriented towards research and technological development. The general objective of the project is to obtain *dynamically reconfigurable optical-wireless backhaul/fronthaul with cognitive control plane for small cells and cloud-RANs*.

1.2.1. Project fundaments

The 5G-XHaul project works with the latest technologies and paradigms, such as small cells, Cloud-Radio Access Networks (C-RAN), Software Defined Networking (SDN) and Network Function Virtualization (NFV) with the aim of providing broadband connectivity in cost efficient and flexible networks, oriented towards the future fifth generation of cellular networks. Dynamic environments both in the backhaul and fronthaul architectures become a must, and 5G-XHaul

proposes a converged optical and wireless network that can connect the access and core network.

Lots of partners (from companies to research institutions or university research groups) are contributing in:

- Introducing advanced millimetre Wave and optical transceivers.
- Development of international standards through technical and economic contributions.
- Development of converged optical-wireless architectures and network management algorithms for mobile scenarios.

1.2.2. Thesis contribution

The thesis being presented here is framed in the last of the points analysed in the previous section, as we will be working with network management algorithms in cellular networks. The main contribution of this development towards the 5G-XHaul project consists in studying the traces from a real network placed in the centre of Athens and property of COSMOTE, evaluating its behaviour and generating network models that can predict several parameters of the network depending on previous values.

With the simulation environment that has been built, we have been able to test some of the proposed network management algorithms, and evaluate the results in terms of user connectivity and energy saving.

1.3. Document structure

This document is divided into several chapters and sections in order to explain the full development of the project in a thoughtful way. First of all, an introduction to the project and the description of the main objectives have been presented. Chapter 2 presents some explanations about the theoretical basis required to understand the project, regarding both cellular networks and data analysis tools, explaining also some of the state of the art of the combination of both fields. In Chapter 3, the reader can find the main information about the tools being used for the development, as well as the data sets that were available. Chapter 4 is devoted to the main results obtained in the two big parts of the project, as explained in previous paragraphs. Finally, some conclusions and the direction of future work is presented, alongside with the bibliography that has been consulted during the project, some abbreviations and acronyms used in the report, and additional information that is collected in the annexes.

CHAPTER 2. THEORETICAL BACKGROUND

2.1. Mobile technologies

It is important to highlight the architecture of the networks that will be studied in this thesis. As introduced in the previous section, we will be mostly working with real LTE networks, although the expansions towards the future generation of networks is worth being considered, so the basics about those standards are summarized here.

2.1.1. LTE and LTE-A

LTE (Long Term Evolution), commercially marketed as 4G[1], is the 3GPP standard for high-speed wireless communications proposed in their Release 8, and based in the structure of legacy GSM, EDGE, UMTS and HSPA technologies. It is packet-oriented and designed to support high traffic loads and speeds reaching peaks of 150Mbps in the downlink and 75 in the uplink. Moreover, it was later powered up with some improvements in the subsequent Releases 9 and 10, the latter being called LTE-A (the real 4G, according to ITU-R), with an A for Advanced.

In this thesis, we are not so much interested in the functionalities and capabilities of such networks, but rather in their architecture, consisting of an access network, the E-UTRAN (Evolved-UMTS Terrestrial Radio Access Network); and the core network, the EPC (Evolved Packet Core). We will focus in the access part.

The E-UTRAN is built with base stations called eNB (evolved NodeB), which are capable of managing the access network without the need of a Radio Network Controller, present in previous generations. eNBs are interconnected using an interface called X2, which allows for faster handover performance and enhanced Radio Resource Management functionalities. eNBs may also have different sectors and/or cells (it is interesting to remember that sectors are not directly bound to cells, as each cell has a different carrier frequency, and a sector may operate several frequencies and therefore several cells).

In Fig. 2. 1 we can see a simplified version of the E-UTRAN, with the S1 interfaces connecting to the EPC, consisting on the Mobility Management Entity (MME) and the connection to the outer network through the Serving and PDN Gateways (S-GW and P-GW, respectively).

[1] Despite the commercial name, it did not achieve the requirements specified by Radio department in the ITU (International Telecommunication Union) to be part of the real fourth generation.

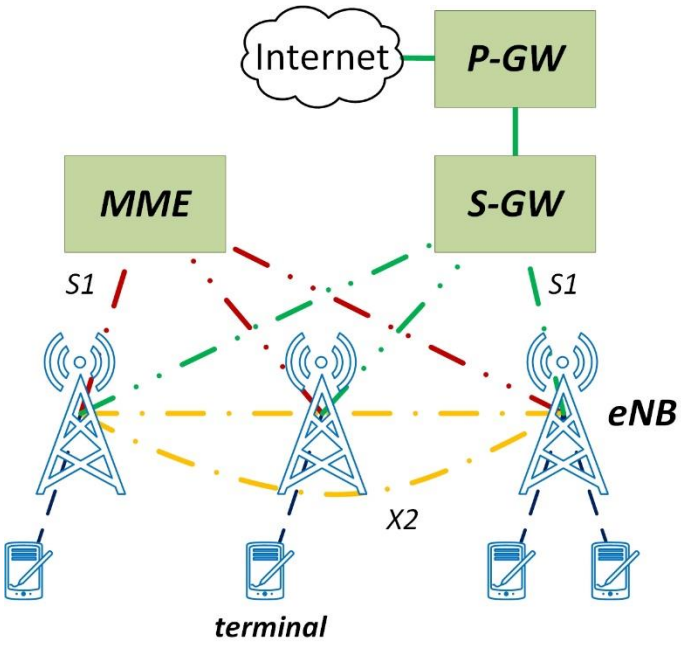


Fig. 2. 1 E-UTRAN and EPC in an LTE network

2.1.2. 5G

The fifth generation of networks, despite not being yet fully standardized, introduces an interesting change in the paradigm of the network architecture. The trend of 5G networks seems to be the implementation of Software Defined Radio (SDR) architectures where the intelligence, which may be running over a Software Defined Network (SDN) platform, is separated from the RF unit.

Wireless base stations will first have a baseband processing unit, called BBU (BaseBand Unit), which will be placed in some centralized equipment room, possibly with other BBUs, forming a BBU-pool. These devices will be connected to the RRH or RRU (Remote Radio Head, also known as Remote Radio Unit) via fronthaul links (including both optical fibres and wireless links). BBUs will be highly customizable and upgradable, and are in charge of most of the functionalities of the network, besides those related to the physical transmission of the signal, which are derived to the RRH.

Meanwhile, SDN (as defined by the Open Network Foundation) seems another interesting paradigm for the management of future networks. SDN provides a dynamic environment in which the network can be easily and automatically reconfigured thanks to the decoupling of the control plane (where decisions about traffic forwarding are taken care of) from the data plane (traffic forwarding). SDN integrates really well with SDR thanks to the hierarchical architecture that it introduces, as it is divided into three main layers (*Applications* in charge of the network management decisions, the *Controller* programming the network devices, and the *Network devices* themselves, the dumb infrastructure in charge of the forwarding plane) interconnected through totally programmable interfaces (a northbound interface that is, in general, an API (Application Programming Interface); and a southbound interface that communicates the controller with the devices, and generally is managed by the OpenFlow protocol).

As a conclusion, we can say that current and future networks have been evolving towards a combination of different techniques that may vary from fully distributed scenarios where the intelligence resides in the base stations (eNBs) and network terminations must coordinate to provide an optimal performance; to centralized intelligent devices that manage a network consisting of *dumb* terminations that are only in charge of the physical radio functionalities. One of the hot topics in this field consists on minimizing the power consumption in such networks, maybe using the collected data to study the behaviour of the network and improve the efficiency in their operation. There exactly is the point in which this thesis is focused.

2.2. Machine Learning

This section is devoted to provide some understanding on what *machine learning* techniques are, how can one work with that -relatively new- paradigm, and what type of results can be achieved with different problematics.

2.2.1. Concept

The field of computing in charge of studying how computers can learn on their own with the minimum human interaction has seen a great increase in popularity in the last decades. It is also true, though, that generally there are slight misunderstandings between similar concepts, and despite being used for the same purpose, each of them is specialized in a different area, and have a varying scope depending on how vast is the field they cover. In the following paragraphs, we will try to shed some light on those concepts.

2.2.1.1. Artificial Intelligence

Artificial Intelligence (AI) is a broader, older concept that aggregates all the knowledge related to how machines can have some kind of autonomous intelligence that provides them with the skills to work as a human would.

The classical understanding of AI works over the *knowledge based approach*, which tries to extract rules that represent some simple knowledge that can be reproduced. For example, playing chess is a task that can be based on knowledge, as a machine just needs to know the allowed movements for each piece and the objective of the game. Meanwhile, there are more complex tasks that are difficult to develop using simple rules, such as recognizing an object in an environment, as the object can be in different positions, sizes (further or closer to the camera), etc.

AI is used for general concepts such as speech recognition, planning or problem solving, although for specific tasks in which the volume of data cannot be handled or the problem is too complex to be solved with rules extracted from the current knowledge, the basic approach of AI is not enough.

2.2.1.2. Machine Learning

Machine Learning (ML) is a new approach under the umbrella of Artificial Intelligence that suggests that machines should be able to learn by themselves. Instead of providing them with repeatable knowledge, we should just give some raw data from which certain patterns can be extracted and learn from them as a human mind would.

The concept of Machine Learning was coined by one of the pioneers of AI, Arthur Samuel, in 1959, although one of the most widely used definitions is the one presented in [4] by another American computer scientist, Tom Michael Mitchell, on a book published on 1997, which says: "*A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P** if its performance at tasks in **T**, as measured by **P**, improves with experience **E**.*". In easier words, experience (i.e. input data) in a certain task is used to learn how to work with a certain problem.

The difference between traditional AI and ML can be easily understood with the following example: it is not the same to program a code that teaches a robot how to walk, or to program one that lets the same robot learn to walk from its own experience (using, for example, statistics of forces applied on each leg, direction on which it fell, stability…).

ML is a very wide computing field that has been under study for the past years and it has reached a peak in popularity recently, with lots of companies of all type devoting resources to research in this area. There are many different approaches to ML, depending on the available data and the problem we want to solve, but that will be studied in depth in section 2.2.2.

2.2.1.3. Big Data

Enterprises of all type have insane amounts of data from which interesting knowledge can be extracted. Every transaction, customer profile or client behaviour can be gathered with others of the same type to extract information that can be useful for the company itself or for others. For instance, the click stream on a web page, the feeds provided by a certain topic in Twitter or the aggregated power consumption in the power grid of each neighbourhood in a city, are reliable sources of data that, studied and processed in the correct way, can result in the transformation of unstructured and scattered data into a summarized specific piece of knowledge.

The Computing Community Consortium (CCC) is an association whose objective is to offer guidelines for the computing research community, articulate different perspectives and interact with policymakers, governments, the industry and the public. It revived the concept of Big Data (concept already coined in 1997 by astronomers Cox and Ellsworth) on 2012 with a White Paper [5] in which the main things about what Big Data is supposed to be are summarized and offers a pipeline about how this process should work (Fig. 2. 2).

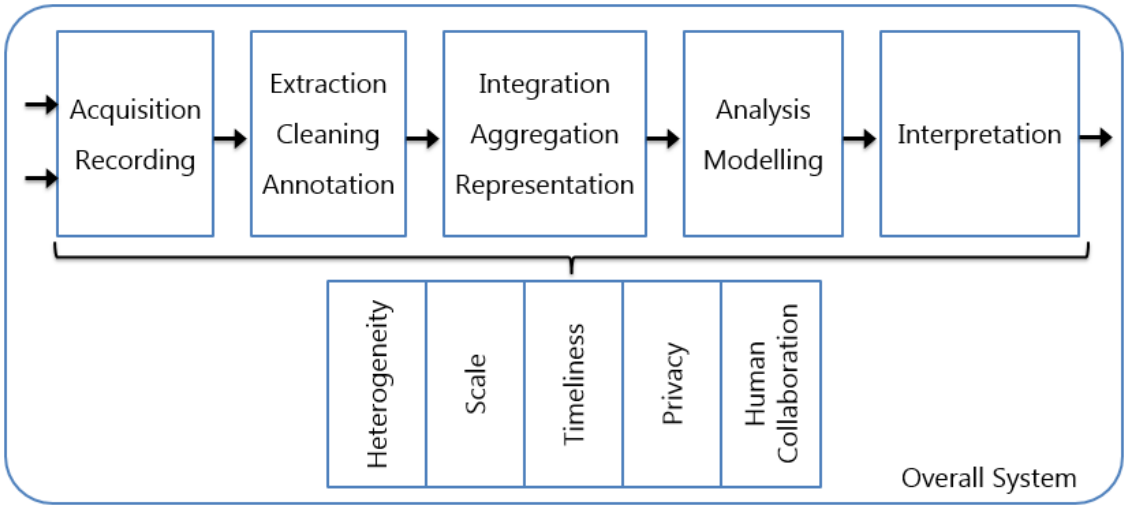


Fig. 2. 2 Reproduction of the CCC Big Data Pipeline presented in [5]

So, as seen in the presented pipeline, it is a general structure of how to obtain information from several inputs, based on the following steps:

- **Acquisition and Recording:** obtain the data from a reliable source and apply the appropriate filters so as to delete that part of the data that is not interesting (take into account that we may be talking of large amounts of data, in the order of TeraBytes or PetaBytes) without discarding useful information. It also comprises the process of generating the appropriate metadata to describe the type of data being recorded and how it was recorded and measured.
- **Extraction and Cleaning:** we need to *extract* structured information from the raw data collected and recorded, and *clean* it so that no outliers (missrecorded or out-of-pattern information that will worsen the obtained model) or errors are passed to the following steps in the chain.
- **Data Integration, Aggregation and Representation:** *integration* consists on providing automatized tools that are capable of producing computer-understandable data from what we already have. Then, data must be *aggregated* by similarity, putting together everything that can or must be processed in the same way. And finally, depending on the type of data being processed, it is also important to find a proper *representation* scheme in which aggregated information can be easily visualized.
- **Data Analysis and Modelling:** the noisiness and heterogeneity properties of Big Data makes it necessary to find adequate querying and mining techniques to extract the information we really need from the vast amounts of data that are available. Performing some *analysis* over the data, different *models* can be obtained, which lead us to the last step.
- **Interpretation:** at the last step of the process there is generally a decision-maker that is in charge of interpreting the results of the whole process and offer a final solution that can be easily understood by the user.

Nevertheless, the Big Data pipeline is constrained by the main challenges also presented in the diagram, which are: **heterogeneity** (Big Data is generally highly heterogeneous and incomplete, so it generally must be cleaned, corrected and completed), **scale** (CPU cores parallelism and cloud computing are two new

paradigms about which Big Data engineers did not have to worry in the past), **timeliness** (sometimes we need immediate results from the analysis process), **privacy** (both for the public and for companies, it is a big concern how their data is being treated) and **human collaboration** (instead of building completely autonomous systems, it might be interesting to have a human in the loop, who might be able to solve some simple problems that are difficult to devise by a machine).

Also, as defined by Oracle, a leading company in the software industry, in [6] and [7], Big Data is defined by the four Vs:

- **Volume:** companies and institutions own large amounts of data, however, it is generally unstructured and its properties are unknown. Depending on the type of data being treated (text files, graphs, images…) and the size of the Data Base (DB), the proportions of the data may vary in a big scale, from some gigabytes of text files to hundreds of petabytes in images.
- **Velocity:** depending on the application given to the collected data, the operation velocity and technique may vary. From data from several months written to disk and analysed once the experimenting period is ended, to real-time applications where it may be streamed directly into memory so as to process it on the fly.
- **Variety:** data has to be different enough so as to provide information. Entropy (as defined by Shannon, it is the amount of data contained in a message) is found in unstructured data that shares enough features so as to be considered of the same type, but is sufficiently uncorrelated so as not to have overlapping knowledge.
- **Value:** nearly all data has intrinsic information that is, in most of the cases, yet to be discovered. There are many examples in which data was already being collected for other purposes, but was not exploded to obtain additional information. This *value* can be extracted using different **Data Mining** techniques. This term is generally comprised under the Big Data one, although being completely strict on their definitions, Big Data refers to the sources of information themselves, while Data Mining refers to the processes applied to that unstructured data so as to obtain knowledge.

Big Data and Data Mining have suffered a huge increase in popularity in recent years, because with relatively small effort (in most of the cases the data was already collected, so we “just” have to study it) really good results can be achieved. Moreover, the cost of storage and computing resources has drastically dropped in the last decade, so it is becoming easier and easier to store enormous databases and process them using efficient techniques. There are also some companies that are beginning to specialize in offering services related to this field, such as Amazon with its elastic computing platform (EC2 [8]), Microsoft with Azure [9], OpenStack [10], etc.

2.2.2. Algorithmic

Machine Learning techniques are based in different types of algorithms, and the decision to use one or another depends, basically, on the type of data we have and the type of output we want to obtain. The objective of this subsection is to

present all the algorithms that may be used at some point during the development of the project, as well as some other useful tools. There are many other algorithms and techniques that cannot be included in this text due to extension restrictions, so this thesis report is limited to those techniques that have been somehow considered for the development of the project.

2.2.2.1. Unsupervised algorithms

Unsupervised learning consists in obtaining some function that can describe the structure of some unlabelled data, i.e. we just offer some raw data to a given machine (which, of course, should have previously gone through the first steps of the Pipeline in Fig. 2. 2) which will provide us with a function to group the data into different sets with certain similarities.

Probably, the most popular unsupervised technique is **Clustering**. Clustering uses ML to group (or technically speaking *cluster*) data into several entities considering the similarity of the features that data has. Generally speaking, clustering is difficult to evaluate, as there is no labelled data set with which to compare results. Each data sample is described by a vector of *N* features *x* that can be used to represent it in an *N*-dimensional space. The distance metric used to determine the closeness between points is really important, as it can drastically change the results. In most of the cases, Euclidean distance (the distance in a straight line between any two points in an Euclidean space) is used, although other more complex techniques can also be applied.

Some of the most common clustering algorithms are:

- *K-Means:* the algorithm groups the data samples into *k* clusters. To start, each cluster selects a random *centroid* and data entities are associated to a cluster according to the shortest distance to each of the centroids. Then, the centroid is moved to the central point (mean value) of the cluster, and the new distances between entities and centroids are computed. The process keeps iterating until each of the data entities is closer to its centroid than to any other, resulting on something similar to this:

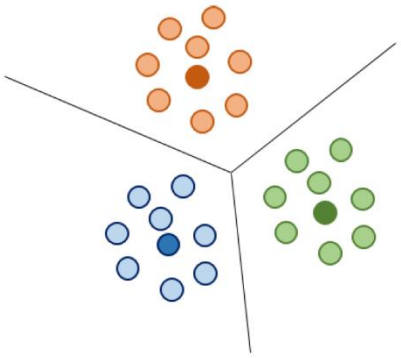


Fig. 2. 3 K-Means clustering result on a 2-dimensional space

- *Hierarchical Clustering:* each data entity represents its own cluster, which is in the following iteration merged with the closest cluster. With this technique, we can cluster data into *k* clusters without previously knowing this value *k*, as the algorithm will try to find the optimal solution. We have an example in Fig. 2. 4.

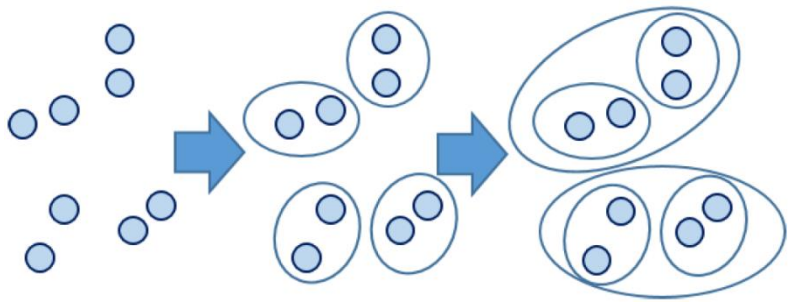


Fig. 2. 4 Hierarchical Clustering example

2.2.2.2. Supervised algorithms

Supervised learning techniques are those that include some labelled feature on each data entity. Data sets used with this ML technique include a set of features, which are the variables considered on each sample, and generally a single labelled feature, which can be a figure (generally in regression) or a text (generally in classification) and identifies the most important variable of the entry, the one that identifies it.

The general idea is to split the data set into some training and test set (the first one is used for training the model that is generated, while the second one is used to test that same model and check the results), using one of the multiple available options: naïve (just splitting the dataset by some percentage), cross-validation (split the dataset into $K = M + N$ equally-sized folds, using M for the training, N for the test and then cycling over them until we can get the mean and standard deviation of each of the folds), nested cross-validation (an even more complex yet effective cross-validation technique), etc.

Then, we can compare the obtained results with the real values. To do that, we input the test set into the model deleting the *label* column, and then we just compare the suggested label with the real one.

Depending on the problem to be faced, we can use different solving paradigms:

- **Regression:** use data with known values to predict the labels of other similar data entities. Regression identifies real numeric values using a function $y = f(x)$ that labels new data entities y (i.e. predict the behaviour of a given feature for a given new data entity) according to the model obtained from all the previously known feature vectors x. Some specific regression algorithms are:
  - *Simple Linear Regression:* use a simple linear function such as (X) to predict the value of $f(x_i)$.

$$f(x_i) = b_0 + b_1 x_i \qquad \text{(X)}$$

  - *Ridge Regression:* use a more complex function to predict a value $f(x_i)$, also considering samples from different time instants t in $x_{i,t}$.

$$f(x_i) = b_0 + b_1 x_{i,1} + b_2 x_{i,2} + b_3 x_{i,3} \qquad \text{(X)}$$

- *Support Vector Machine (SVM):* it is a variation of ridge regression in which the loss function (the function that defines the error committed in the assignment of the label as compared to the real value) is modified. SVM is one of the most popular algorithms, and can be also used as a *classification* technique.

– **Classification:** classification techniques assign a label (from the *N* ones available) to an unlabelled data entity. All entities from the test set end up with a label that can be later compared to the real ones and measure some typical statistics:

Table 2. 1 Confusion Matrix

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (**TP**)	False Negative (**FN**)
Actual Negative	False Positive (**FP**)	True Negative (**TN**)

- *Accuracy:* calculated as in (X), provides the percentage of accurate predictions.

$$\alpha = \frac{TP + TN}{TP + TN + FP + FN} \quad \text{(X)}$$

- *Precision:* calculated as in (X), provides the percentage of guessed values for each class.

$$\rho = \frac{TP}{TP + FP} \quad \text{(X)}$$

- *Recall:* calculated as in (X), measures the fraction of guessed instances over the total instances.

$$r = \frac{TP}{TP + FN} \quad \text{(X)}$$

- *F1 score:* calculated as in (X), is a combination of both precision and recall.

$$F1 = \frac{\rho \cdot r}{\rho + r} \quad \text{(X)}$$

– Some popular classification algorithms are:
  - *SVM:* the same as in regression, but now providing a given class as an output.
  - *Decision Tree:* a sequence of branches is defined; at each intersection, a given function is calculated and, depending on the result, one or other branch is followed. In Fig. 2. 5 , an example tree can be seen, which can be used to classify people depending on several features such as their age range, education, activities, etc. Trees have different properties, although one of the most important may be the total depth of the tree, i.e. the amount of levels of branches it has plus the root. In this example, the tree in Fig. 2. 5 has a depth of 4. This parameter is important because deeper trees may be more accurate, but having really deep trees may lead to

overfitting to the training set, which is an undesired side-effect; on the other hand, fewer levels is more simple to compute and execute, but may be too generic. Imbalanced Data Sets (DS), which are those where a class highly predominates over the rest or has many less samples than the others, can be solved by weighting the samples to balance the algorithm.

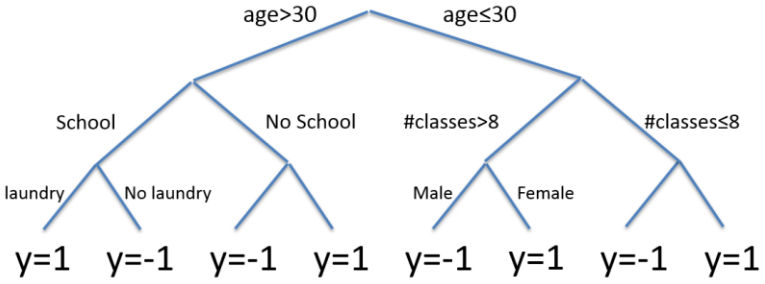


Fig. 2. 5 Decision Tree example

- *Random Forest:* a random forest works as a set of *N* different trees that are run with the same input data and that may produce equal or different results. The output of all the trees is then combined (using different techniques, such as majority voting, average of the results, etc.) and a single solution is provided. The *random* part about these forests is that all trees are different thanks to using different random seeds for their generation, and the diversity of the trees can be configured by tuning some parameters.
- *One vs. all:* multi-class technique that extends any binary classification algorithm to support more than two classes at the same time. It works by performing the function $y = f(x)$ over all the possible classes and comparing the *y* results to see which is the class it has more confidence in.

2.2.2.3. Reinforcement learning – Neural Networks

Reinforcement learning is the last of the main ML paradigms considered for the development of this project. This set of techniques consists in the fact that the system interacts with the environment with a feedback loop, so that each output has an effect over the current model that produced it.

Neural Networks, are based on the methodology of reinforcement learning, by imitating simplified biological learning models such as neurons. Neurons are very simple computational units that produce an output according to some stimulus. The interconnection of an extremely large amount of neurons with high connectivity, plus the capacity of working in parallel, can produce amazing results in the field of ML.

Neurons, as represented in Fig. 2. 6, are computing entities with several inputs x_N and some links w_N that connect and weight each input. It also has a bias value *b* used to correct any possible error, and a single output *y*.

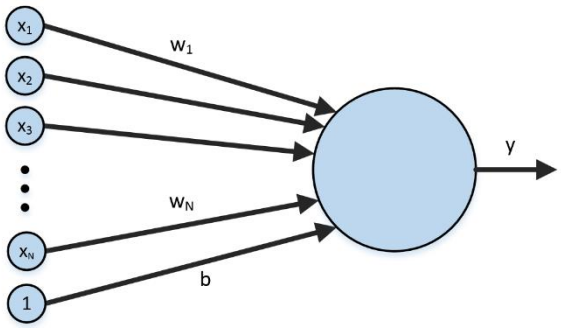


Fig. 2. 6 Example of a Neuron in ML terminology

Neurons can train themselves using the real and desired output depending on different approaches that have been relieving each other all along their history. For instance, the first approach, the Hebb Rule (1949) [11], reinforced those outputs that were correctly predicted; the Perceptron (1950) [12] is just the opposite, using Negative Learning (reinforce the errors); and AdaLine [13] reinforces both cases with an error *e* that is measured as the difference between the desired and real outputs.

Up to now, however, we have only seen how a lonely neuron works. Neurons can be combined to form a neural network that can produce way better results. We can even add hidden layers to form deep neural networks (as in Fig. 2. 7) that can solve any kind of problem, although their understanding and design is out of the scope of this project.

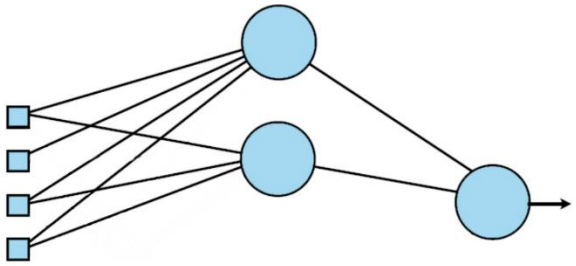


Fig. 2. 7 Example of Neural Network with 1 Hidden Layer

2.3. State of the Art

The objective of this section is to present the state of the art related to the main topics covered in this project. First of all, we give details of the latest investigations about the application of ML to network management, and then we review the literature on power management in mobile networks.

2.3.1. Machine Learning techniques for network management

2.3.1.1. Traffic profiling in radio networks with Machine Learning

As already commented and presented in [11], BigData technologies will play a really important role in the deployment of the future 5G networks, as they are able to transform vast amounts of data into something valuable that can be used, for example, to improve the performance of the network. This paper intends to develop intelligent systems capable of studying the environment, identifying patterns and working with an AI-oriented network control plane. It focuses on using **classification** to identify patterns in the cells deployed in the network and classify them in a set of known classes. Two main use cases are presented, although only the first one is relevant to our project:

- Use case 1: energy saving by means of switching off cells that carry very little traffic at given times of the day. Cells are classified into two classes (candidates to be switched off or not). The input vector to the machine learning algorithms will have 21 components (7 days of the week, 3 time intervals, with the average normalized traffic of the cell), and each of the 4 classification techniques used is tuned manually so as to achieve the optimal solutions. The classifiers will work with 419 cells that offer measures every 15 minutes, and the traffic is measured as the average number of users in the cell with an active data session. Only sets of 10-200 cells are used for the training, and the best results are provided by **SVM** and **Neural Networks**, as measured by the percentage of total coincidences between the classification tool and the category assigned by the expert validation. Section 5.2 in [11] presents the optimum parameter values for each of the classifiers.
- Use case 2: there are several new considerations of LTE regarding the spectrum usage, such as using unlicensed bands (LTE-U) or sharing the spectrum with a primary user (within certain limits). In this case, cells are classified into two classes (candidate cell to boost capacity through additional unlicensed spectrum or cell that does not need the boost). Now, traffic is not normalized (the absolute value is also important), and only 16 components (daily basis, 1 component per hour from 6am to 10pm) are used, with the same classification tools as in the previous use case.

There is another project in [18] whose objective is to estimate the load of a given group of base stations just from the monitoring of the load of a subset of those. The accuracy on the predictor will vary on the base stations chosen for the monitoring (amount and type). To do so, OLS (Ordinary Least Squares) is considered, a linear **regression** technique that, despite being popular, has poor accuracy, reason why an alternative is presented: Lasso (Least Absolute Shrinkage and Selection Operator). The data used for the prediction is formed by

TCP/UDP flows from 400 base stations along 5 days (weekdays). As the variation in traffic patterns all along the day is really high, they divide the data and create models for individual periods of 4h, thus getting 6 separate models for each day.

In conclusion, good results are obtained, both with classification techniques (SVM and Neural Networks) and regression (improved OLS). They are all supervised algorithms, which implies that the training set has to be assessed by an expert, who would manually identify the example cells as one or other class.

2.3.1.2. Using clustering to group entities with similar characteristics

Grouping cells in a network by similar features is another interesting technique that is being exploited in the literature. In fact, most of the sources consulted perform some type of unsupervised clustering algorithm to subdivide the traffic profiling problem into smaller parts.

In [15] two different clustering methods are used, **K-means** and **Gaussian Mixture Models (GMM)**. K-means is a simple method that works well for low dimensional data (while it gets slow for large data sets), which may be able to find local (but not global) optima and has problems with outliers and noise. In this case, they cluster the users, not the cell types, and the simulation that is run is based on the following items:

- Model based on an American city with a city centre and high population density, surrounded by a highway and a suburban area with low population density.
- 11 base stations, 33 cells, 600 active users.
- 60 minutes with 100ms time step.
- Use of several type of users (indoor, low mobility, medium mobility, high mobility).
- Users downloading 1MB files.

The features are similar to the one in the real COSMOTE data sets we have available for the project, and with the provided results, we can conclude that user classification may not be a promising solution for the problem being boarded.

Tree classification models are able to perform automatic variable selection, being this an advantage if we don't know which features (if any) are more important than others when performing the classification. However, Trees tend to result in a quite low accuracy, reason why Random Forest are used instead. Random Forest works with several decision trees that are aggregated together so as to improve accuracy, being also more robust to overfitting, as each tree will work with a different part of the original training data, and there's always a factor of randomness involved.

Using the clusters identified in the clustering process, they train and test the classification Random Forest algorithm. They also study the importance (Gini index) of each of the **5 variables** that are being taken into account (time to handover, user throughput, CQI, number of active users and cell throughput), and how the behaviour of the classifier would change if one of those statistics went missing. The results obtained in [15] prove that training on a 1/60 of a time frame still offers accurate classification results for the rest of the time frame.

There is another really interesting research in [16] that introduces new concepts not seen in similar papers. It investigates the prediction of hourly traffic volumes based on historic data, with the goal of provisioning an energy efficient system. In order to remove redundant information, obtaining only the stable information, wavelet decomposition is used, as well as other techniques:

- **K-means clustering** method to automatically classify data into K groups.
- **Wavelet transform**, a data processing tool that works at different frequency scales or resolutions. This procedure helps dividing data into more stable and tractable sets, which can then be processed individually, obtaining a higher overall prediction accuracy. Moreover, this method ensures orthogonality, which will help with the predictions.

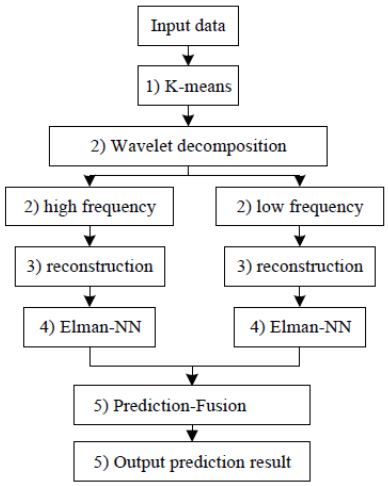


Fig. 2. 8 Forecasting framework proposed in [16]

- As the prediction algorithm, an **Elman Neural Network** is used. It is a powerful tool for predicting time sequences, based on an input layer, a hidden layer and an output layer, with a fourth context layer that feeds back the state of the outputs of the hidden layer without weighting. The problem presented in this project is highly nonlinear, reason why ENN was chosen as the method for the prediction, as its ability to store internal values helps in the approximation of nonlinear dynamics.

Some results regarding clustering and framework (Fig. 2. 8) accuracy are presented in Fig. 2. 9:

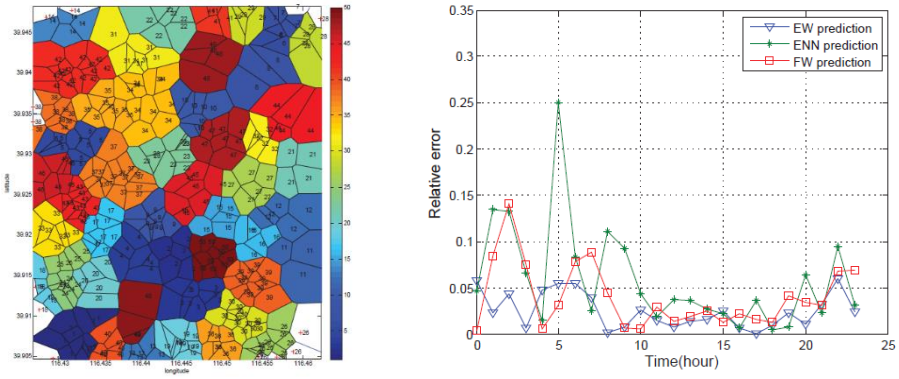


Fig. 2. 9 Clustering results (left) and traffic load prediction relative error (right) achieved in [16]

Finally, in [17] a project is presented whose main aim is to identify clusters of cells with similar 24-hour traffic profiles. To do so, besides removing some outlier cells (158 from the original 2175 cells), the data was cleaned so as to take only the relevant information (**hours 8 to 23, only weekdays**, as the rest didn't follow any pattern that could be modelled). The clustering process helped them with the classification of cells into 6 different groups, which lead to an improved estimation of the required capacity between a 25 and an 80% (as seen in Fig. 2. 10).

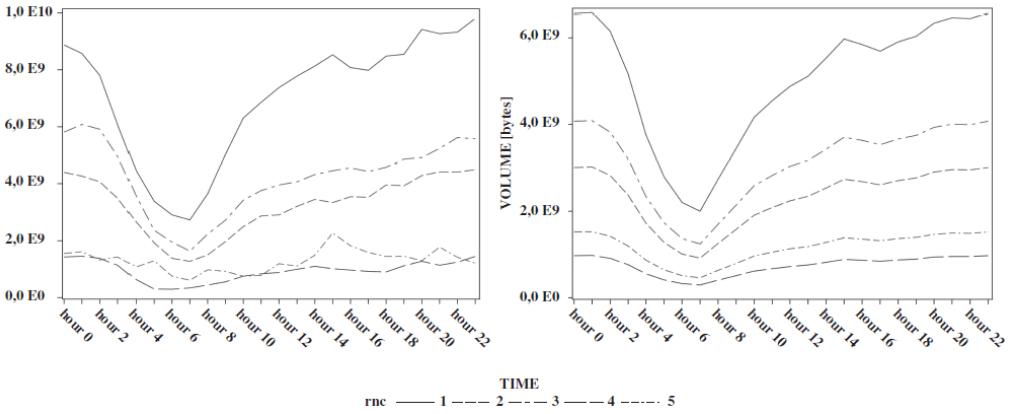


Fig. 2. 10 Real traffic (left) and predicted by the cluster-based model (right)

2.3.2. Power management in mobile networks

As for the project developed during this thesis, it is also interesting to take into account the literature regarding power management in radio networks, as one of the applications is to provide energy efficiency to an access network scenario.

[19] discusses the performance of a scenario where the cells with idle capacity requirements are turned to sleep mode. Based on other references, this paper evaluates the potential performance of an energy-saving scheme where underutilized cells are put to sleep, as LTE eNBs have a significant power saving when working on idle mode, as can be seen in Fig. 2. 11:

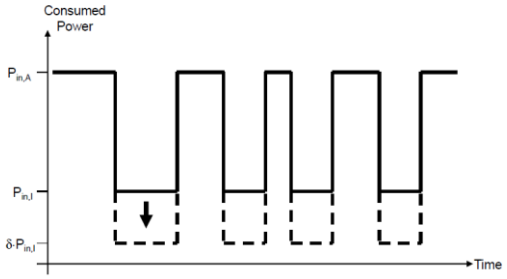


Fig. 2. 11 Assumed model for the eNB power consumption over time [19]

$P_{in,A}$ represents the power consumption in active state, $P_{in,I}$ in idle state, and $\delta \cdot P_{in,I}$ is the result of using fast cell DTX (Discontinuous Transmission) [20], where $0 < \delta < 1$ and in the paper is generally assumed to be $\delta = 0.1$.

Meanwhile, [19] studies different network deployments and cell DTX to reduce power consumption, with the results summarized in Fig. 2. 12. The left graph shows which network distribution achieves lower consumption, while the right one represents the consumption over a day, and the **heavy energy saving at night**, during the low-traffic hours, where a reduction of up to 49% is achieved, given that during that period of time, cells are in **idle mode for 98%** of the time. This reinforces the hypothesis presented on this thesis about considering valley hours as the reference for cell type clustering and switch-off algorithms.

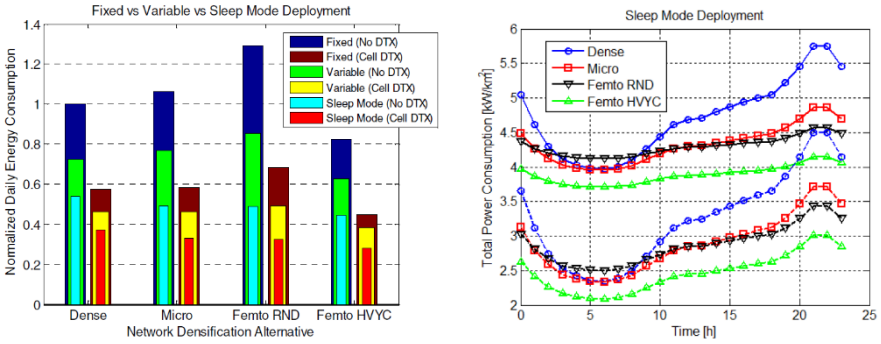


Fig. 2. 12 Normalized daily energy consumption for the different network densification alternatives (left) and total power consumption of the sleep mode deployment as a function of time without fast cell DTX (solid curves) and with fast cell DTX (dashed curves) (right)

Finally, [21] presents their own joint access-backhaul SDN platform for efficient management of a radio network. The research devoted to energy efficiency uses a similar technique to what is originally thought for this project, i.e. turning on and off access nodes dynamically to achieve energy saving, and they also highlight the importance and consequences of eliminating access nodes from the network. To do so, they run an algorithm that, besides identifying the potential candidates to be switched off, it adds three Quality of Experience (QoE) constraints:

- Network coverage: failure probability that a user loses connectivity.
- Admission control: blocking probability for flows with dedicated bandwidth.
- Delay for best-effort flows: flow delay exceeding a given threshold.

So this paper raises the problematics that appear when switching off some access nodes of the network, which is something that has to be considered.

CHAPTER 3. TECHNOLOGIES AND ALGORITHMIC

This chapter is devoted to the description of the main software, technologies and data sources used for the development of the project. Starting with the 5G-XHaul project and the provided data sets, we later go into using RapidMiner for Machine Learning purposes, and finally a brief explanation of the main functionalities of the simulator designed for the thesis.

3.1. 5G-XHaul

As explained in section 1.2, this thesis is framed under the 5G-XHaul project, and therefore we have had access to real data from UMTS and LTE networks from COSMOTE[2], a Greek mobile telephony company. This section is devoted to explaining the characteristics of the networks from which we had accessible data, and also describing the datasets we were provided with.

3.1.1. Real scenario

The datasets provided by COSMOTE include 17 BTS located in centre Athens, 10 of which have both LTE and UMTS cells, while the 7 remaining only support UMTS. The approximate location of the base stations (BTS) from which we have data is depicted in Fig. 3. 1[3].

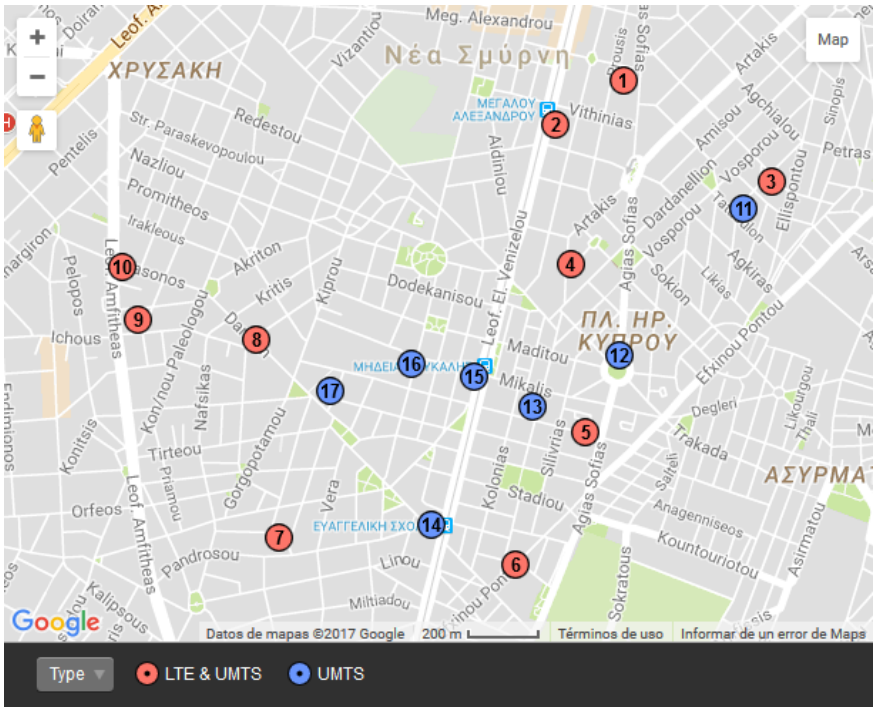


Fig. 3. 1 COSMOTE network BTS location

[2] https://www.cosmote.gr/hub/
[3] https://es.batchgeo.com/

Athens is a heterogeneously dense city which, in an area of around 750 km^2 (including dense urban, urban and suburban areas) has a deployment of over 600 LTE eNodeBs. Note that, on the same area, second generation, UMTS and HSPA+ sites are also deployed, but our interest is focused on LTE. With such figures, the average density in the city is about 0.8 eNBs/km^2, although there are some additional considerations in scenarios with a maximized density:

- In really dense areas, such as the city centre, there are more than 30 eNBs with over 70 sectors in 1 km^2, which is makes a density of 10 eNBs / km^2.
- In suburban areas, the density may be around 2.5 eNBs/km^2.

In those deployments, LTE eNodeBs are configured to offer a coverage range that moves between 100m and 600m, corresponding to different type of cell deployments depending on the geographical area being covered. The power consumption of those eNodeBs are unknown figures, so we will assume the values given in [19] (Table 3. 1) as a coarse approximation. *PeNB,max* is the maximum eNodeB output power, while *Pin,A* and *Pin,I* correspond to the consumed power per cell in active or idle mode, as represented in Fig. 2. 11.

Table 3. 1 eNodeB power consumption model from [19]

eNodeB Type	$P_{eNB,max}$ [W]	$P_{in,A}$ [W]	$P_{in,I}$ [W]
Macro RRH	40	336.3	238.4
Micro	1	152.4	129.3
Femto	0.1	16.6	14.4

The sites available in the COSMOTE network have from 1 to 3 cells and, in general, they have a bandwidth of either 20 (in most of the cases) or 10 MHz.

3.1.2. Data sets

For the development of this project, we were provided with 2 packs of data sets. They are two independent data sets covering 15 days each; the first of them is on the last trimester of 2015. In both cases, we have 15 days of data statistics collected every 15 (for the LTE cells) or 60 minutes (for the UMTS cells), including two weekends, where the traffic profiling may be different.

As for the statistics collected during each quarter hour from the LTE cells, they are written on a spreadsheet with different pages, one for each BTS, and there can be found many different metrics. The ones that have been studied in deeper detail for the development of the project are those that are underlined. The rest were considered finally not as relevant, and there are some more metrics that do not cover any important feature considering the development.

- ***Time:*** time variable including the date and time of the sample.
- ***Cell Name:*** text variable including the ID of the cell with the format XYLTE, where X is the BTS ID and Y the cell ID inside the BTS.
- ***Data Traffic (MB):*** UL/DL data traffic in MB.

- ***Mean Cell Throughput (Mbps):*** average LTE UL/DL throughput in Mbps.
- ***Max Cell Throughput (Mbps):*** maximum LTE UL/DL throughput in Mbps.
- ***Average # UEs:*** average number of UEs each TTI (Transmission Time Interval).
- ***Max #UEs:*** maximum number of simultaneous UEs in a TTI.
- **Total UEs in eNB:** total UEs in eNB during the whole hour.
- ***Average PRB Usage per TTI (%):*** UL/DL average Physical Resource Blocks usage (i.e. PRBs Used / PRBs Available) per TTI.
- **Intra eNB Latency:** Intra eNB latency.
- **Average CQI:** average Channel Quality Indicator (CQI).
- **CQI Distribution per Level (00-15) (%):** CQI distribution per level for each of the 16 CQI values as percentage of total samples.
- ***MCS Distribution per Order (%):*** % distribution of Modulation and Code Scheme (MCS) usage (Low (MCS0-9), Medium (MCS 10-19), High (MCS20-28)).
- **MCS Distribution per Scheme (MCS0-MCS28) (%):** MCS distribution per Scheme for each of the 29 values as percentage of total samples.
- **Average PUCCH SINR:** average PUCCH SINR.
- **Average PUSCH SINR:** average PUSCH SINR.

3.2. RapidMiner

This section is devoted to explaining the tool with which the first half of the project has been mostly developed. In the first steps of the thesis, there has been a big involvement with data analysis and identification of predictive models for the COSMOTE traces.

The tool chosen for the development of this progress is RapidMiner [22], and the aim of this part of the chapter is to familiarize the reader with RapidMiner and to give a basic how-to so as to understand the incoming sections that go deeper into the understanding of this software.

3.2.1. What is RapidMiner?

RapidMiner is an integrated data analytics platform that offers a unified environment for machine learning-related projects. The platform has three main products: Studio (the one used in the project and that will be commented in-depth later), Server (a platform to deploy online projects from Studio) and Radoop (a Hadoop [23] implementation for RapidMiner).

RapidMiner Studio is the tool that has been used for the project and, as defined in the official website, it is a “powerful visual workflow designer for rapidly building predictive analytic workflows. This all-in-one tool features hundreds of data preparation and machine learning algorithms to support all your data science projects”. So, in short, it offers a graphic interface in which, by means of boxes that perform different functions, we can program data analytics projects.

RapidMiner Studio offers a free version with a limitation of 10.000 rows of input data (which in general, for a relatively small project, might be enough, although we have worked with some larger datasets in some particular experiments). It also has an Educational program with unlimited dataset size, and further licenses with improved performance capabilities.

3.2.2. How to

RapidMiner offers an easy-to-use difficult-to-master platform that, by means of being visual, is accessible for anyone without even basic programming skills. It also offers a huge variety of extensions (either created by the community or by the developer team) that can be purchased, and for this project, the Series Extension pack [23] has been key.

In the screenshot Fig. 3. 2, we can see the RapidMiner Studio workplace, with some information about the different parts of the software: *1* are the toolboxes, *2* shows the datasets and the projects that have been imported to RapidMiner, *3* includes all the modules available and a search box to find them easily, *4* is the main process window where the project is developed (*4a* are the inputs of the project and *4b* the outputs that will show once the program is executed), *5* are the options available for each of the modules (boxes) in the project, and *6* is a help window that explains the functionalities and options of each of the modules.

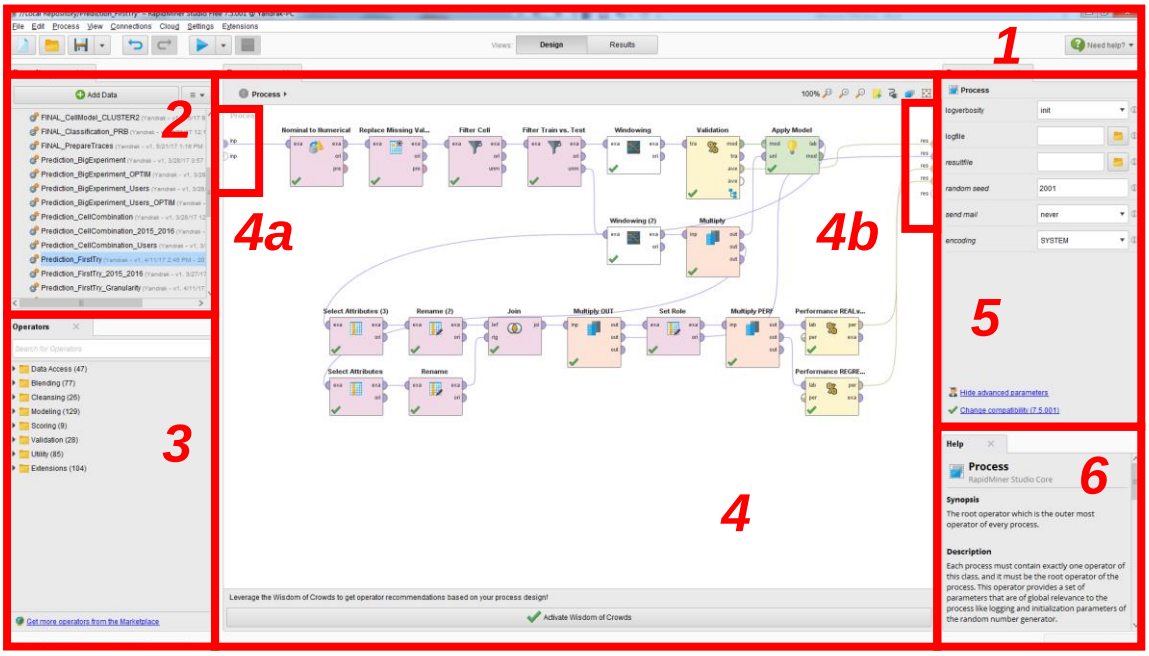


Fig. 3. 2 Screenshot of the RapidMiner Studio workplace

RapidMiner offers a really wide variety of operators that can be further complemented with free and paying extensions. These are some of the most important sets of modules:

- **Data access:** import data from different sources (files, databases, cloud, Twitter feed...).

- **Blending:** modify the data set by changing some values or filtering attributes.
- **Cleansing:** adapt the dataset by performing multiple functions such as normalization, replacing missing values, getting rid of duplicates or outliers, etc.
- **Modelling:** it offers up to 129 different algorithms for regression, segmentation, weight calculation, optimization of parameters…
- **Scoring and validation:** finding the final performance results, as well as the confidence on them and additional cross-validation modules.
- **Other:** execute external scripts or macros, log information, generate random data, divide processes…
- **Series Extension:** the additional Series Extension is needed to work with time series, which is what is being done in the project, predict a future sample from the preceding ones. This extension offers the *Windowing* module, which basically stacks *N* samples in a single row so that all the parameters from those *N* samples can be considered part of a single prediction. An example is shown in Fig. 3. 3. The Series Extension also offers specific metrics designed for time series.

Sample	A	B
0	10.3	26.2
1	15.0	35.3
2	9.6	20.4
4	…	…

Sample	A - 2	B - 2	A - 1	B - 1	A - 0	B - 0
0	9.6	20.4	15.0	35.3	10.3	26.2
1	…	…	…	…	…	…

Fig. 3. 3 Dataset modification performed by the *Windowing* operator with *window_size = 3*

Just to end with this section, below (Fig. 3. 4) the reader can find a brief explanation of how a simple prediction project works.

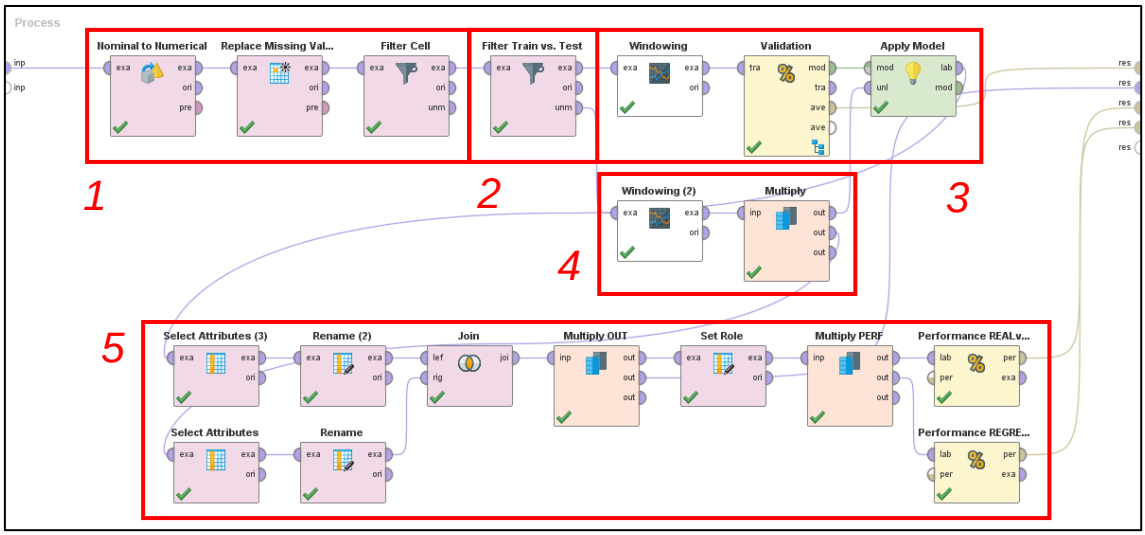


Fig. 3. 4 RapidMiner example project: predicting mean cell throughput

The figure shows an easy example of how to predict the parameter *Mean Cell Throughput* (cf. section 3.1.2) on a given cell. It works as follows: after the input file is chosen (in our case the dataset for eNB1) *1* is in charge of pre-processing

the data by cleaning it and filtering only the cell we are interested in (in this example, Cell 1C); *2* divides the training and test set by date, i.e. the first 12 days are used for training and the last 3 days are the test set; *3* is in charge of creating the regression model that, by using an SVM algorithm, will predict the next sample based on the previous ones; *4* generates the appropriate test set from the data; and finally *5* cleans the output data so that it can be represented in an understandable way, while also providing some additional metrics about the accuracy of the model.

The prediction output of the project above can be seen in Fig. 3. 5, and it also outputs a performance accuracy of 0.708.

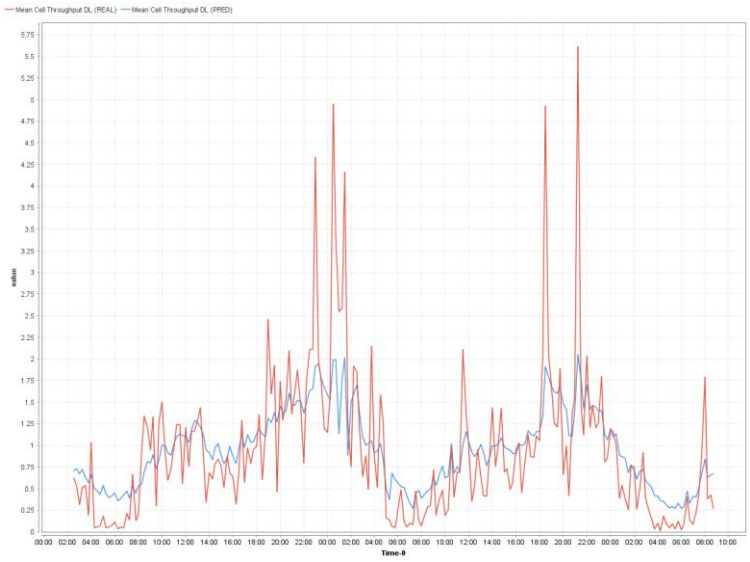


Fig. 3. 5 Output prediction of Fig. 3. 4; in red the real values, in blue the predicted ones

3.3. Working on realistic scenarios: simulation

Once the first part of the project regarding data analysis and traffic profiles modelling was ready, the project required some additional work to test everything that was learned during the previous stages. In this section, an overview of the simulation environment used for the last part of the project is presented.

3.3.1 Tools and environment

With the objective of evaluating different management strategies in a cellular network, the idea of building a simulator capable of generating realistic scenarios from the real traces in the data sets came up. It should fulfil the following requirements: it had to be realistic and based on the COSMOTE traces, it had to be fully customizable and upgradable according to the new ideas and project perspectives that could appear, it must collect statistics from the randomly generated scenarios, and it could include some graphical resources to back the results with an easier interface.

Finally, and after some thought on the matter, Java was the language chosen for the development. Java is an object-oriented, concurrent, class-based programming language developed within the WORA (Write Once, Run Anywhere) paradigm, in such a way that a compiled Java piece of code can be executed on a Java-driven platform without any further recompilation. Moreover, it includes some easy-to-use graphical interfaces thanks to *Swing*[4], a Graphical User Interface widget toolkit available for Java.

3.3.2 Simulator structure and functionalities

The custom Java-based simulator has more than one thousand lines of code spread into 5 classes, as illustrated in Fig. 3. 7:

- ***simulator.java:*** main class, where the simulation runs.
- ***ModelFromTraces.java:*** program designed to extract a simplified CSV model from the real COSMOTE traces. It basically takes the samples from the 15-day-long dataset and generates, for each cell, 3 CSV files (one for the MCS distribution model, another one for the throughput model, and the last one for the PRB usage model) with 96 lines, each of them referencing a time sample (24 hours x 4 samples per hour = 96). Then, on each line, there are all the samples taken at the same time of the day in the 15 days, producing something similar to what is seen in Fig. 3. 6.

```
6  0-29.6,26.3,30.7,26.2,23.3,19.2,33.5,26.6,29.1,31.5,29.0,21.9,26.2,36.9,39.5
7  1-18.6,14.8,21.6,18.4,28.5,29.6,21.3,30.0,22.3,20.3,18.0,34.2,31.6,27.8,28.3
8  2-16.9,12.4,21.2,21.6,17.2,23.3,24.2,19.2,19.3,20.5,25.3,21.1,21.0,21.8
9  3-17.8,8.5,17.9,17.6,18.8,26.5,28.3,24.4,13.5,15.5,19.3,23.7,31.3,23.1,20.5
```

Time	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample 6	Sample 7	Sample 8	Sample 9
0	29.6	26.3	30.7	26.2	23.3	19.2	33.5	26.6	…
1	18.6	14.8	21.6	18.4	28.5	29.6	21.3	30.0	…
2	16.9	12.4	21.2	21.6	17.2	23.3	24.2	19.2	…
3	17.8	8.5	17.9	17.6	18.8	26.5	28.3	24.4	…
4	…	…	…	…	…	…	…	…	…

Fig. 3. 6 Example of PRB model from Cell 6B extracted using *ModelFromTraces.java*

- ***Cell.java:*** class that defines the *Cell* object, and includes as parameters: position coordinates in the simulated scenario, instantaneous throughput, PRB usage and MCS assignment (high, medium or low), list of attached UEs, cluster to which it belongs, cell being followed by the traces, state (ACCEPT/ON, OFF or FULL), ID, BTS to which it belongs, cell type (Macro, Micro or Femtocell) and *E* (the user migration penalty).
- ***BTS.java:*** class that defines the *BTS* object, which only includes the coordinates in the scenario, the ID, and the 3 cells that belong to that BTS.
- ***UE.java:*** class that defines the *UE* object, including its position in the scenario, the instantaneous throughput, PRB and MCS, its ID, whether it is served or unserved, and ρ (the share of the total throughput of the cell that it has).

[4] http://docs.oracle.com/javase/tutorial/uiswing/

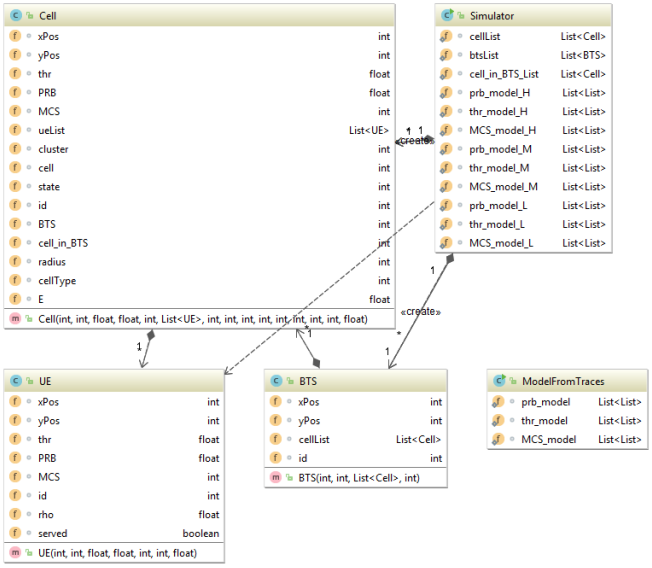


Fig. 3. 7 Simulator class diagram

3.3.2.1. Simulator operation

The simulator has its main execution line and some complementary functions to obtain extra functionalities and statistics. The main line has some previous steps:

- Cluster the real cells into three groups: high, medium and low traffic load.
- Generate the models for each cell and parameter: different **.csv* files for each of the metrics (PRB, UE, THR and MCS) and load (very high, high, medium) containing a compacted version of the metrics.

Then, the program runs as follows:

I. Load the PRB, THR and MCS model for each of the clusters.
II. Generate the scenario with the following characteristics: 2000 x 2000 m size, 30 randomly placed trisectorial eNodeB (i.e. 90 cells).
    a. The scenario is divided in a 6x5 grid so as to simulate a more realistic scenario, which provides a similar distribution of cells to the observation in centre Athens. Each square in that grid has a size of 333.3 x 400 m, and a single eNodeB is randomly located in an inner area of 233.3 x 300 m, as in Fig. 3. 8, to make sure most of its coverage is within the limits of the scenario (Fig. 3. 10).

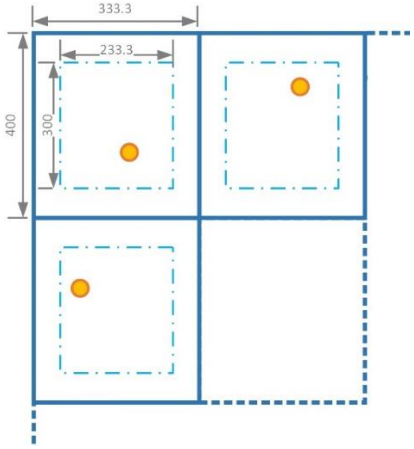


Fig. 3. 8 Random distribution of eNodeBs in the simulation scenario

b. 3 cells with equal radius are assigned to each eNB. The radius is set by a random uniform distribution between 300 and 600 m. Depending on the radius, a different cell type is assigned (ranges 300-400, 400-500 and 500-600 correspond to ***femto***, ***micro*** and ***macro*** cells, respectively).
c. Each cell is assigned a cluster with a certain probability (in the real scenario, there are only 2 cells belonging to the high load cluster, 7 to the medium load, and 18 to the low load, so the probability for each cell to be in a given cluster are, respectively, 2/27, 7/27 and 18/27.
d. Each cell is assigned a real cell belonging to the same cluster. A first sample is taken randomly from all the samples in the first time instant (i.e. first row as shown in Fig. 3. 6). For example, SimCell 1 is assigned to cluster 1 (medium load) and to RealCell 5, so one sample among the fifteen available in the first time instant of RealCell 5 is assigned as the initial value of PRB, THR and MCS for SimCell 1.
e. A given number of UEs is generated on each ON cell according to the traces, with a random position inside the cell sector (i.e. users must be in the coverage radius of the cell and the appropriate angle, as in Fig. 3. 9).

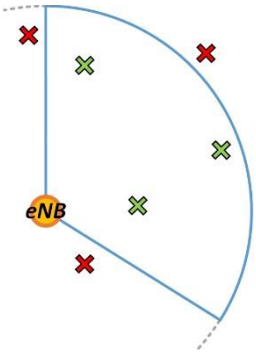


Fig. 3. 9 Random distribution of UEs in a cell in correct (green) and incorrect (red) positions

f. Each user is assigned a random share of the throughput and PRBs configured to that cell.
g. Each cell is assigned a migration penalty parameter *E* used for the calculation of the increase in PRBs when a user migrates from one cell to another. E is calculated as in (X), where *PRB* is the total load in the cell in %, w_i is the weight (1, 2 or 3, depending on the MCS high, medium or low) of each of the *N* associated UEs and t_i its throughput.

$$E = \frac{PRB}{\sum_1^N w_i \cdot t_i} \qquad \text{(X)}$$

Finally, the simulation generates a scenario like the one in Fig. 3. 10.

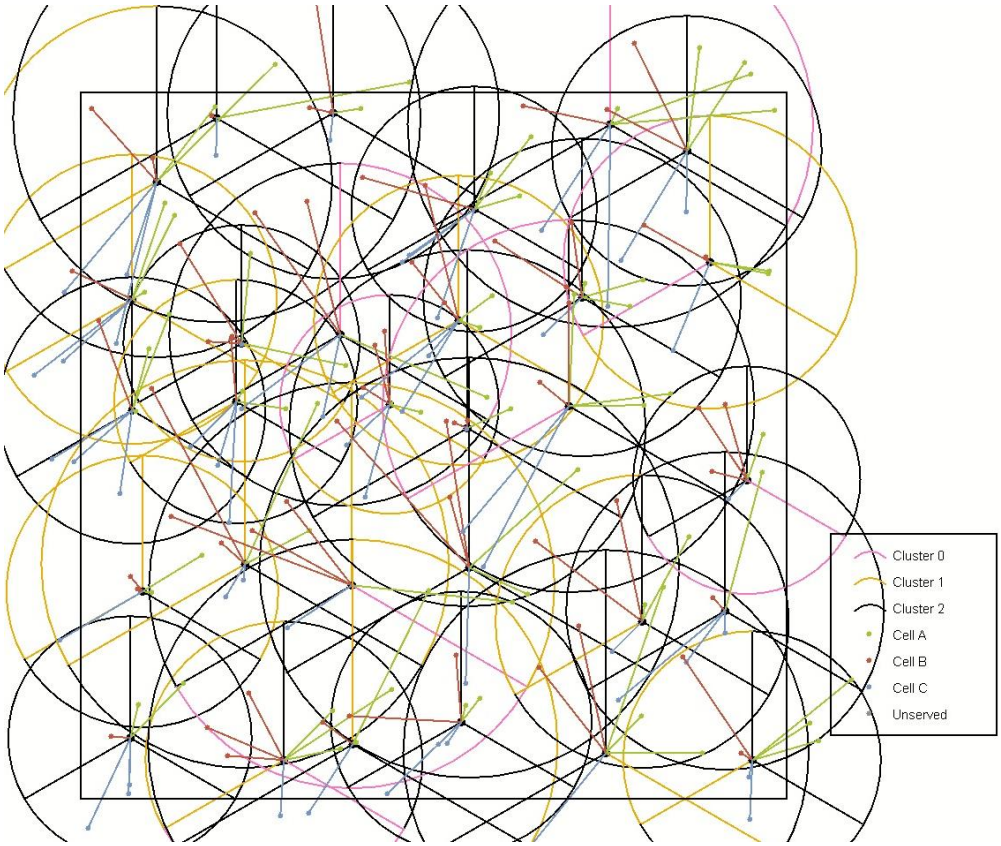


Fig. 3. 10 Random scenario generated by the simulator

III. On each further iteration (one iteration are 15 minutes of the simulation, so to run one full day, we will need 96 iterations), a new value for PRB, THR and MCS is obtained, according to (X), being PRB_{new} the new randomly selected PRB sample for the next iteration taken from the same real cell (RealCell 5 in the previous example) and *a* the tuning parameter, being here equal to *0.3*, value obtained from testing with different figures and choosing the one that generates the most realistic synthetic trace. A new set of UEs is also generated.

$$PRB = PRB \cdot a + (1 - a) * PRB_{new} \quad \text{(X)}$$

IV. After each iteration, record the traces of all the cells on independent *CSV* files.

V. Once the simulation is finished, obtain statistics about:
   - ON/OFF cells on each iteration.
   - Power consumption on each iteration.
   - % of unserved and migrated UEs on each iteration.

3.3.2.2. Main simulator functionalities

Besides the main execution line, the simulator also has some added functionalities, used for different purposes, but all of them enhance the experience of working with it. The main additional functionalities are:

Calculate cell density

It is interesting to study the cell density in the scenario, i.e. for each point in the 2000 x 2000 surface, calculate how many cells serve it, and then obtain the average. In order to obtain some valuable results, the simulation should be executed several times, and then study the most interesting values, such as the average and the maximum and minimum cell density achieved with the chosen distribution.

The information obtained with this functionality is useful to understand to how many neighbours, on average, a user that is left unserved by a cell that is being switched off can migrate.

Migration

Another key feature of the simulator consists in migrating UEs from one cell that is being turned off to another. The process works as follows:

I. Identify the state of each cell as *ACCEPT*, *FULL* or *OFF*. *ACCEPT* implies that the cell has some spare capacity to allocate new UEs; *FULL* implies that the cell uses its PRBs over a certain threshold and, therefore, should not receive more UEs; *OFF* means that the usage of the cell is below another threshold and can be considered as candidate to be switched OFF to save energy.
II. For each cell, check if it has been switched off and, in such case, migrate each of its connected UE individually. To do so:
    a. Find the best eNB to the given UE (i.e. the one that can be accesses using the highest MCS) that is in state *ACCEPT* (if there is no *ACCEPT* covering that UE, move to the best *FULL*).
    b. Calculate the PRBs that the UE will consume in the new cell. Note that if the assigned MCS is worse than the previous one, more PRBs will be needed to maintain the same offered traffic.
    c. Add the UE to the new cell.
    d. Repeat until all the UEs from the cell being switched off have been migrated or remain in an unserved state (if no other cell can allocate it).
III. Turn the cell off.
IV. Count the amount of migrated UEs and the amount of unserved UEs after the migration.

Fig. 3. 11 shows some examples of migrations done in a smaller scenario. In the first couple of images, we simulate that the bottom cell in the upper eNodeB is being turned off, and therefore the connected UEs (represented in blue) have to migrate to the nearest cell, which happen to be the red and green ones from the second eNodeB, respectively. On the other hand, the right graph shows an unsuccessful migration, where the migrated cell is the red one from the first eNodeB and, as a result, both UEs end up being unserved.

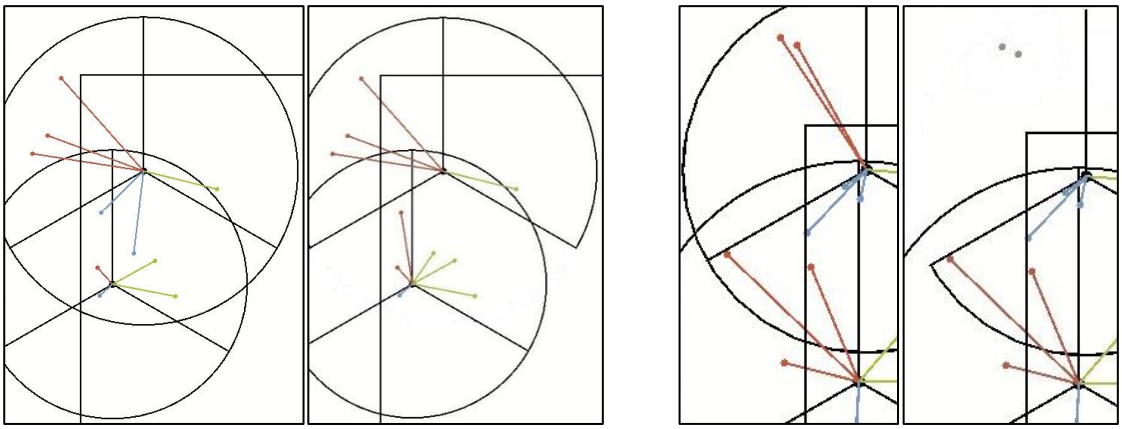


Fig. 3. 11 Migration of the users in the south cell in the top eNodeB (left) and unsuccessful migration with users that end up being unserved (right)

Print results

On each iteration of the simulator, we can call a function that prints a summary of the traces for each cell, as depicted in Fig. 3. 12. It shows information about all the cells, including: position, total throughput and PRB usage, state (0, 1, 2 correspond to ON/ACCEPT, OFF, FULL), cell type (*macro*, *micro*, *femto*) and UEs connected to the cell.

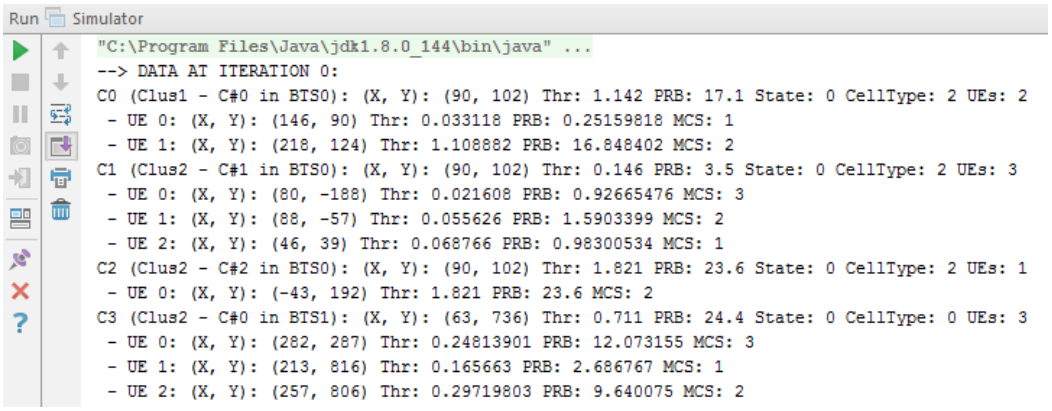


Fig. 3. 12 Traces printed for the first iteration of a simulation

Draw scenario

An additional function has been programmed, which uses Java's Swing GUI to print the scenario running in the simulation at any moment. Fig. 3. 10 or Fig. 3. 11 are examples of the graphical representation. It prints the trisectorial eNBs with the UEs connected to each cell.

Having presented the main tools that have been used for the development of the project, now we can move on to the development itself and the main results that were obtained through the research.

CHAPTER 4. DEVELOPMENT AND RESULTS

4.1. Applying Machine Learning techniques over 5G-XHaul data

Once the main details about the tools used for the development of the project have been presented, it is the moment to present the main results regarding both the 5G-XHaul data analysis and the simulation scenarios. This section is devoted to explain the main experiments performed with the provided datasets from COSMOTE and RapidMiner, classified into the three main Machine Learning techniques: regression to predict the network usage level in the next time sample, classification to predict the state of each of the cells for the following samples, and clustering to group cells according to their main features.

4.1.1 Study cases for the 5G-XHaul data sets

In order to work with the 5G-XHaul data sets, different study cases were prepared. For each of the experiments, an appropriate optimization had to be done, which was relatively easy (although slow) using the "Optimize Parameters" module in RapidMiner Studio. This tool can be configured so that, for a given set of parameters, it tests as many values as required, performing an iteration of the program for each of the possible combinations, and finally providing the best result with the obtained scores and corresponding values for the tested parameters. For example, optimizing three different parameters (*C* [tolerance for misclassification], *ε* [insensitivity constant, part of the loss function] and *ε convergence* [precision of optimization conditions]) the software will execute the program *T* times, being $T = a \cdot b \cdot c$ (where *a*, *b* and *c* are the number of possible values assigned to each tweaked parameter).

The main study cases developed are the following ones:

- Use the full data set (including weekends and weekdays) to predict *T* days using the previous *D* days. Only one cell is used for this prediction, and all the parameters are used.
- Separate weekends from weekdays, in order to see if the traffic pattern is different and if there is some improvement in the prediction when using independent models.
- Combination of all the cells in a sectorized eNB to predict the average traffic in the base station. This way, less computation capabilities are required since we will have one model for any eNodeB. It is important to see whether this simplification worsens the performance in an acceptable margin or if the cells in a base station are different enough so as to have to work with independent models.
- Combination of all the cells from different eNBs, or by cluster (with similar characteristics) to compute a single model.

- Combination of the two available data sets, the one from 2015 and the one from 2016, to see if there are similarities in the behaviour of the cells, and if an improvement in the results is achieved.
- Decrease granularity to 30-minute intervals instead of 15 minutes.
- Use PRB usage to classify the cell into three possible states: OFF, ACCEPT or FULL, so that: $OFF < th_1 < ACCEPT < th_2 < FULL$, where th_1 and th_2 are two manually assigned thresholds that divide the cell state into three different phases as explained in section 3.3.2.2.
- Cluster the cells into different groups automatically depending on the cell features.

In the following subsections, the results of the different experiments are presented, grouped according to the ML technique being applied.

4.1.2 Predicting network usage: Regression

The first approach that came up to design an intelligent algorithm for turning cells on and off depending on the traffic load for each time interval of 15 minutes consisted on predicting the throughput, number of users and PRB usage for the next time sample considering *N* past values. To do so, the SVM algorithm was used, and the optimization tools of RapidMiner were also required to obtain the optimum results in each of the experiments.

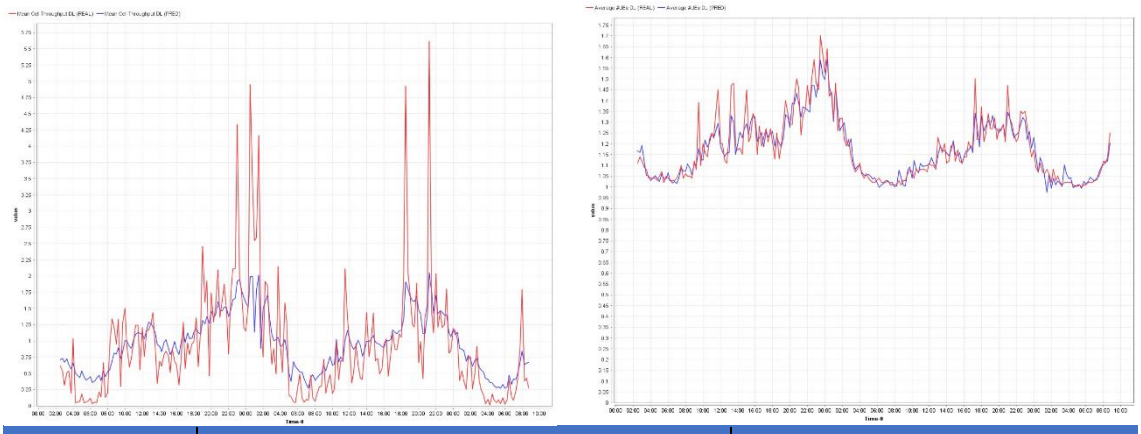
Simple throughput and UE prediction

Some simple initial experiments were performed, predicting throughput and number of UEs in a single cell, being the results presented in the corresponding table. As seen in Fig. 4. 1 and Fig. 4. 2, the prediction of throughput (left) and number of UEs (right) for two highly loaded cells is quite good. In the throughput prediction, the trend is accurate and, in most of the cases we can even identify the high load peaks, although their absolute value is not properly predicted, probably due to the smoothing parameters in the algorithms. That is the reason why RMSE (Root Mean Square Error) and RE (Relative Error) are not considered as metrics to measure the performance in throughput prediction, because the error committed during the peaks was too high.

These two experiments were executed using 12 days as training set and the last 3 days of the data set (the 2016 one) as test set. The metrics used for the accuracy measurements are accuracy over the test set (*Acc* in the tables) and accuracy of the cross-validation over the full data set (*X-Val* in the tables[5]).

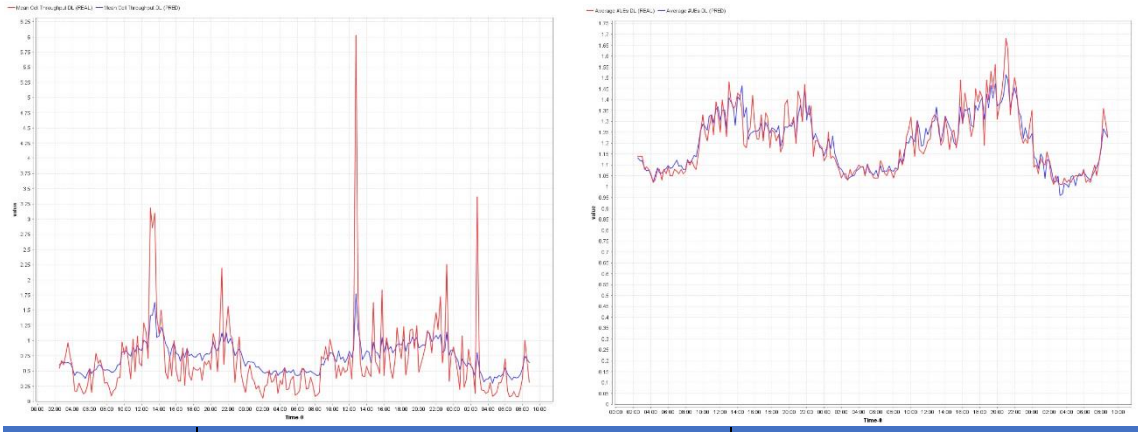
Please, see Appendix A for a larger version of the figures that follow, which have been reduced here due to space constraints.

[5] The software performs some type of cross-validation over the data set, although no specific information about how it works is given by this RapidMiner module



Prediction	X-Val	Acc.	RMSE	RE	C	ε	Window
Throughput	0.662	0.708	-	-	0.001	0.5	10
#UEs	0.522	0.755	0.053	2.99%	0.055	0.001	10

Fig. 4. 1 Regression prediction of Mean DL Throughput and #UEs of 3 days in cell 1C (real values in red, predicted in blue)



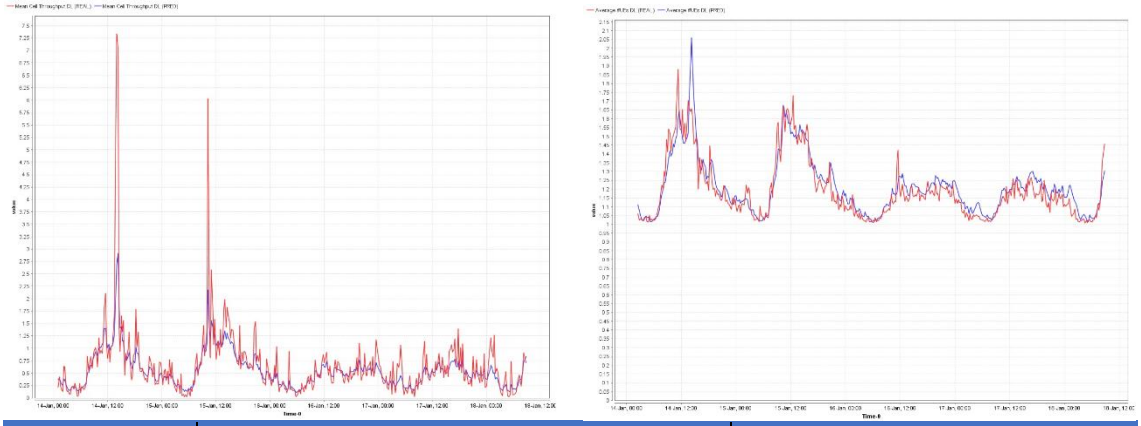
Prediction	X-Val	Acc.	RMSE	RE	C	ε	Window
Throughput	0.657	0.753	-	-	0.001	0.0001	10
#UEs	0.516	0.753	0.049	2.84%	0.0055	0.001	10

Fig. 4. 2 Regression prediction of Mean DL Throughput and #UEs of 3 days in cell 8A (real values in red, predicted in blue)

Traffic aggregation

Another experiment consisted in aggregating all the traffic of a given multicell eNB to have a single model for each of them. The aggregation was done by average value, i.e. if an eNB has 3 cells, the resulting data set is the average of all the parameters of those 3 cells, and the prediction is also done for the average aggregated traffic of the base station. The conditions of the experiment are the same as before: 12 days used to predict the 3 last ones, with parameter

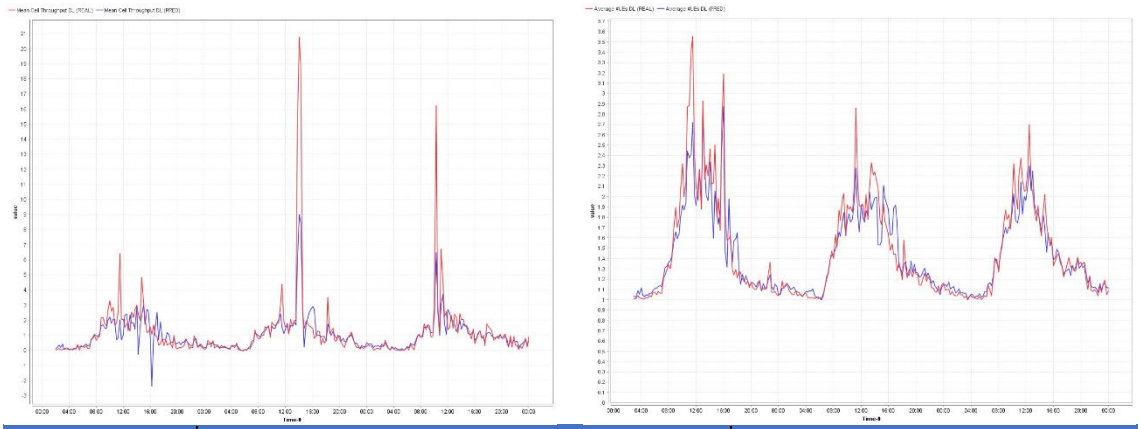
optimization, and it is interesting to notice that, in all the aggregation experiments, the window size chosen is smaller than in the rest of cases (only 4 samples).



Prediction	X-Val	Acc.	RMSE	RE	C	ε	Window
Throughput	0.605	0.778	-	-	0.001	0.05	4
#UEs	0.606	0.554	0.069	4.02%	0.0025	1	4

Fig. 4. 3 Regression prediction of Throughput and #UEs of 3 days in eNodeB 3 (real values in red, predicted in blue)

A different experiment regarding traffic aggregation was also studied: aggregate (average) traffic of similar eNodeBs, generate a single model from those samples, and predict the values of a specific cell. For this experiment, eNBs 1 and 3 (with similar traffic profiles) were chosen, and aggregating the traffic from the 6 cells in those eNBs we could predict throughput and UEs in a specific cell (3A).



Prediction	X-Val	Acc.	RMSE	RE	C	ε	Window
Throughput	0.566	0.763	-	-	0.006	0.0001	8
#UEs	0.522	0.655	0.173	6.01%	0.002	0.001	12

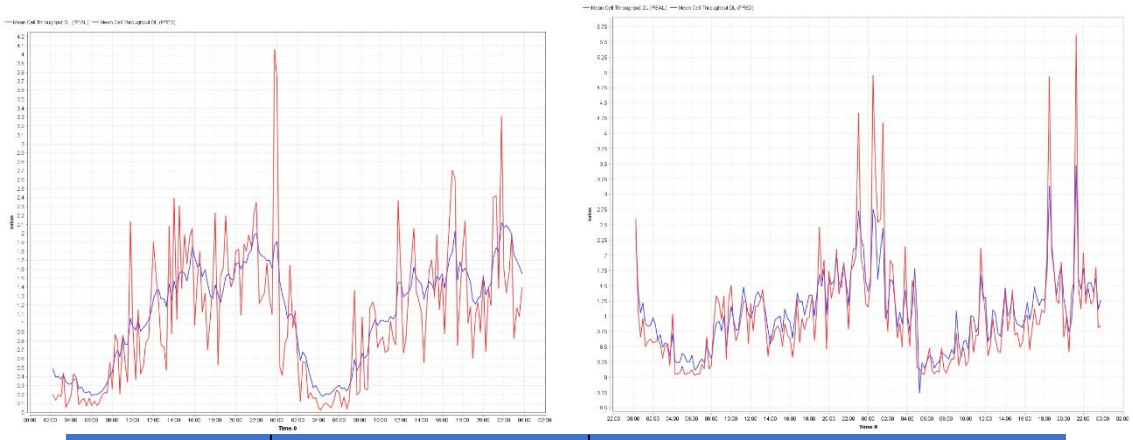
Fig. 4. 4 Regression prediction of Throughput and #UEs using the aggregated traffic from two eNBs (1 and 3) (real values in red, predicted in blue)

The results, as shown in Fig. 4. 4 are not distant from those obtained in the previous cases, leading to the conclusion that we could generate unified models for the prediction, given that the cells are similar enough among them. This allows us to simplify the system by having fewer models, and thus requiring less computation capabilities.

Separate weekdays and weekends

The idea of separating weekdays and weekends is quite logic, and comes from the fact that traffic profiling is different during working days (i.e. Monday to Friday) and weekends. This is also experimented in some of the documentation studied in section 2.3, and the results looked promising. In our case, we could see that predicting throughput differentiating weekdays and weekends, did not bring apparent improvement for the weekdays, although for weekends it resulted in a higher accuracy when studied independently, with a better prediction of the absolute value of the peaks, probably thanks to the fact that the optimal window becomes smaller: only 2 past samples are required to predict the future value.

For the weekday experiment, and considering only the most recent data set, 10 weekdays were available, two of which were taken for the training set, while the 2 remaining were the test set. On the other hand, for the weekends experiment, only 4 days were available, and therefore 3 of them were used for training, while the remaining Sunday was used as test.



Prediction	X-Val	Acc.	C	ε	Window
Weekdays	0.677	0.72	0.0005	0.185	10
Weekends	0.634	0.81	0.01	0.08	2

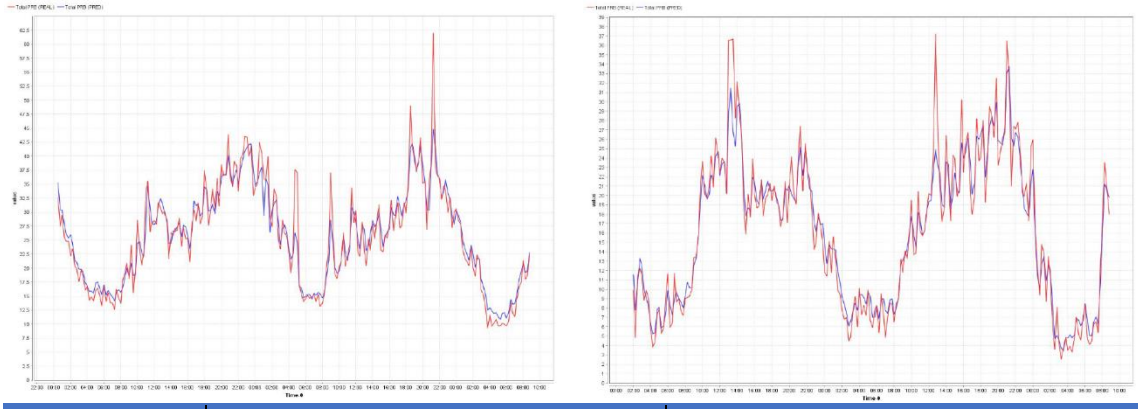
Fig. 4. 5 Regression prediction of Throughput separating weekdays (left) from weekends (right) (real values in red, predicted in blue)

Comparing aggregated data (as in Fig. 4. 4) to the values shown in Fig. 4. 5, it is apparent, while keeping similar scores in the weekday experiment, the weekends do get a higher performance in the metrics.

PRB Prediction

After the initial steps of the development, the idea of using PRB utilization instead of throughput or number of users to determine the load of a cell came up. That is the reason why further experiments with PRB were performed. The metric used in the following cases is the aggregation of the UL and DL PRB utilization, and the latest data set is used too, considering the first 12 days for training and the 3 last for test.

The accuracy of two of the highest loaded cells (1C and 8A), as stated in Fig. 4. 6 is really high, with low error values, a good prediction of the absolute value of the peaks (which was not achieved when measuring throughput) and the valleys, and a really small window of only 2 samples. Those reasons contributed to choosing PRB as the final metric to measure the load of the cells and applying the algorithmic to choose whether they should be turned off or not.



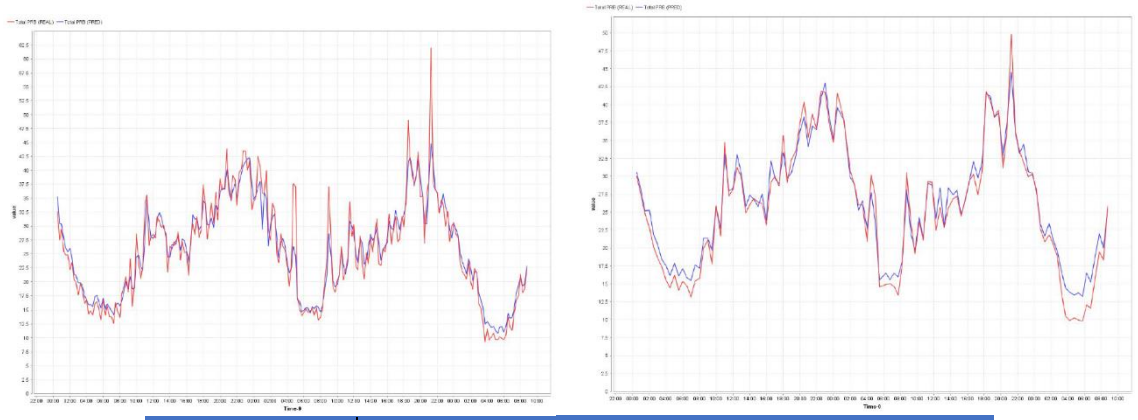
Prediction	Acc.	RMSE	RE	C	ε	Window
PRB (1C)	0.903	2.02	10.91%	0.02	0.0001	2
PRB (8A)	0.812	2.61	7.26%	0.02	0.0001	2

Fig. 4. 6 Regression prediction of PRBs in cell 1C (left) and 8A (right) (real values in red, predicted in blue)

Granularity reduction

Reducing granularity reduces the computing complexity of the problem, as less samples are considered. However, there is a trade off with accuracy, since the predicted sample is obtained less often and therefore, instead of calculating the value of a parameter for 15 minutes, we extend it to 30 minutes or longer.

According to the experiments, the most appropriate granularity change might be from 15 minutes to 30, averaging each sample with the following one, so as to have a dataset with half of the inputs. The results are interesting, as the trend of the different metrics is accurately predicted, and thanks to the smoothing effect of the aggregation of samples, the accuracy is slightly higher.



Granularity	Acc.	RMSE	RE
15 minutes	0.812	2.607	7.26%
30 minutes	0.838	1.754	7.11%

Fig. 3. 13 Regression prediction of PRBs with granularity of 15 (left) and 30 minutes (right) (real values in red, predicted in blue)

Conclusions

After working with lots of different experiments in regression, the following conclusions could be extracted:

- Throughput prediction is accurate, but although the peaks are, in general, properly identified in time, their absolute value in Mbps is not correct, and the curve of the evolution of throughput is smoother than in reality.
- Number of UEs in a cell is not a good metric to consider given that these metrics showed small variations among low number of users.
- PRB utilization, as the aggregate (addition) of UL and DL PRB in a cell seems to be the appropriate metric to determine the state of a cell.
- Aggregating traffic might be a good idea, as it simplifies the future simulation scenario, having fewer prediction models with a really good accuracy, comparable to that of specific models for each cell.
- Reducing granularity is a trade-off between complexity and detail. Being the results similar when halving the granularity, it is a matter of design to choose whether it is better to work with half of the samples or if it is interesting to have information about the network more or less often.
- Separating weekends from weekdays, which was previously studied in the literature, can provide better results under certain conditions too.

4.1.3 Predicting network state: Classification

After the first approaches to cell load prediction using the regression paradigm, an idea came up, consisting on, instead of calculating the exact value of PRB utilization on each cell, simply try to predict the state of a cell (*OFF*, *ACCEPT*,

FULL) depending on the predicted PRB usage for the next time sample.

Classification, as explained in chapter 2, is a supervised algorithm, which means that, in order to classify entries by a given variable, it must have information about that same variable in the training set. That is the reason why, for all the time samples in the data set, a new attribute *STATE* was created, following the pattern *OFF* < th_1 < *ACCEPT* < th_2 < *FULL*, with th_1 and th_2 being different thresholds depending on the dataset being used.

Then, instead of using a regression algorithm, a classifier was used: a Random Forest with 10 trees, depth equal to 10 and window size equal to 1, being these the parameters outputted as optimal by the optimization tool in RapidMiner. In this example, we use th_1 = 7% and th_2 = 25%, and the results are shown in Fig. 4. 7. The left image shows the comparison between the classifier result (which are the straight lines with three levels meaning *OFF*, *ACCEPT* and *FULL* from bottom to top), which shows an accurate classification. The right graph is the same representation after removing the regression curve, so that the errors made in the classification are more clearly visible. As shown in the same figure, the number of errors is low and, in most of the cases, those errors are given between OFF and ACCEPT states, motivated by the fact that during valley hours, the PRB utilization may oscillate often around th_1.

Fig. 4. 8 illustrates the confusion matrix of the results in the previous figure, and shows a really high general accuracy of nearly 95%. Most of the errors, as stated before, in *OFF* cells wrongly classified as *ACCEPT* (8 errors) or the other way around (3 errors, which in fact make the energy saving algorithm more conservative, as we avoid errors of occupied cells being turned off).

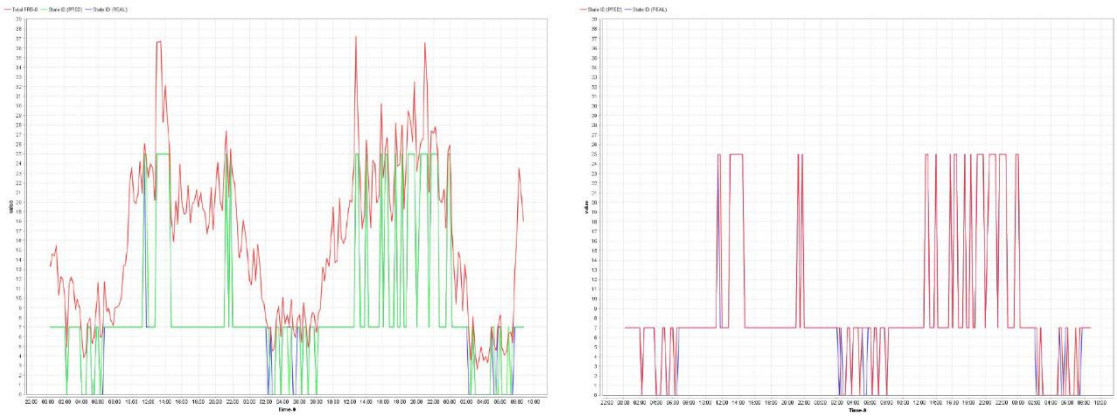


Fig. 4. 7 Classification compared to the time evolution (left) and classification accuracy (right)

accuracy: 94.67%

	true ACCEPT	true OFF	true FULL	class precision
pred. ACCEPT	154	8	0	95.06%
pred. OFF	3	27	0	90.00%
pred. FULL	1	0	32	96.97%
class recall	97.47%	77.14%	100.00%	

Fig. 4. 8 Classification accuracy and confusion matrix

These results show the power of a classifier with properly identified thresholds, and proves that no precise values are required to identify the load of a cell, as we can simply program a classifier that labels the cells into three different groups. In fact, the final simulator program, as will be explained in the following section, uses a tree classifier extracted from the traces generated by the simulator to classify the cells into the three possible states.

4.1.4 Grouping cells: Clustering

As studied in the literature, grouping cells by similar characteristics may be a good technique to simplify some of the problems related to the project. In our case, it might be interesting to group similar cells into several clusters so that multiple cells share the same models for cell state prediction. Moreover, it can be useful in the generation of synthetic cells in the simulator, as we can make a cell to follow the profile of any of the cells in a cluster, introducing some randomness in the simulation.

The clustering process in the project consisted on grouping in three different clusters, which finally resulted to correspond to very high, high and medium loaded cells. Clustering with bigger and smaller *K* (number of clusters) value were also tested, but the most appropriate one seemed to be *K* equal to 3, as in all the cases, the tendency was to group very high and highly loaded cells in 2 different clusters, and then the rest altogether, so testing with a higher *K* only created more clusters in the middle-load class, which was not as interesting. As the project is oriented towards energy saving by means of turning off cells, which is more probable during the night, the differentiation of cells has been done during nightly hours, reason why only the samples between 00 and 08 am have been considered. For the time samples of all cells in that time interval, the average and the variance of PRB utilization was computed, so that 15 values (one for each day in the dataset) per cell were obtained, and finally only two values were considered for the clustering process: the average of averages, and the average of variances.

The final results of the clustering are shown in Fig. 4. 9, where the centroids of the three clusters are represented. The very high load cluster has a high PRB utilization during night, also with high variance, and are cells 1C and 6A, according to Fig. 4. 10. The high load cluster has a moderate PRB utilization but also presents high peaks and valleys during night, while the medium load cluster has low PRB utilization average and variance. The complete table of values for the 27 cells can be consulted in annex BB.1.

Attribute	cluster_0	cluster_1	cluster_2
PRB Avg	8.560	17.199	4.301
PRB var	25.965	61.410	5.412

Fig. 4. 9 Clustering centroid table

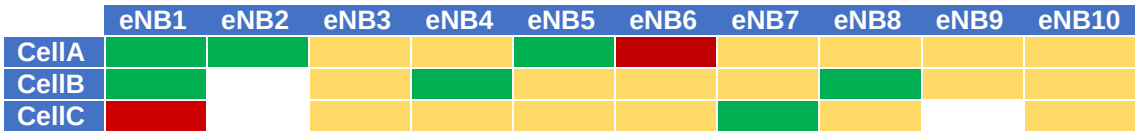


Fig. 4. 10 Cells clustered by load: very high (red), high (green) and medium (yellow)

4.2. Simulation results

The second and last part of the project consisted in, as explained in previous sections, programming a simulator environment with which we can evaluate different resource management strategies in random scenarios created using the knowledge achieved in the first part of the project. In this project, we will test an intelligent system for switching cells on and off dynamically, with the aim of reducing energy consumption.

That being said, the main results obtained with the simulator described in section 3.3 are the ones contained in the following paragraphs.

4.2.1 General statistics

Scenarios, as stated in section 3.3.2, are randomly generated, inspired by the distribution of the real COSMOTE network in Athens. The behaviour of the cells is also adapted to that of the real ones, as extracted from the data sets. Those are the reasons why, on each execution, the results are different, but they all share some similarities, as they are trying to simulate the behaviour of the real Greek network.

Synthetic traces

The synthetic traces define the behaviour of the random scenario and are generated by the simulation engine, inspired by the real COSMOTE traces. It is important, for the sake of the project, to have a realistic scenario that matches the results from the first half of the project, based on the study of the data sets. However, it is also interesting to have scenarios that are different enough, to take profit of the simulator and obtain different results that may apply to the real network.

The graphs in Fig. 4. 11 (which can be consulted in a bigger scale in Appendix C.1) represent the comparison of traces extracted from the COSMOTE data sets and synthetic traces generated by the simulator. Three cells from different clusters have been selected, 1C (very high), 1A (high) and 8A (middle), and they all show similarities with the real cells, but the differences are such that the simulation results can vary from an experiment to another.

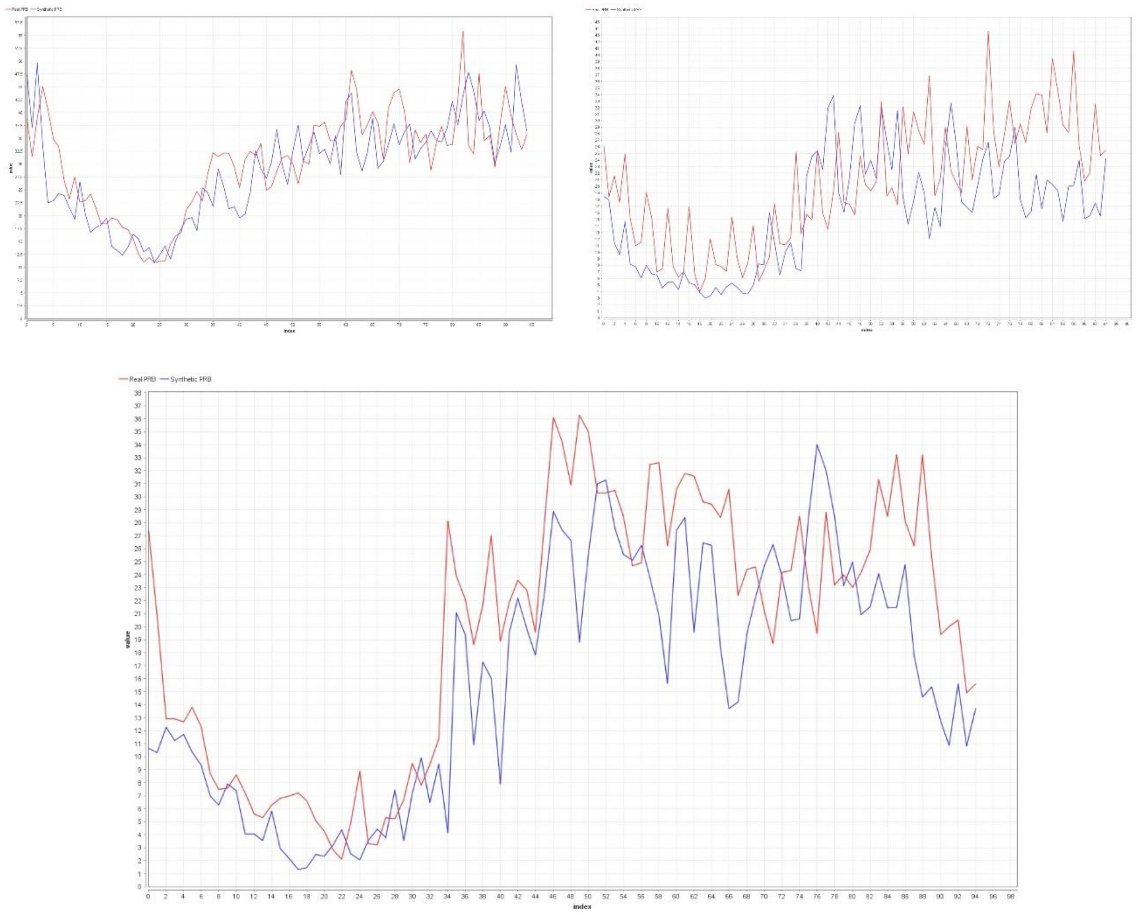


Fig. 4. 11 Comparison of real (red) and synthetic (blue) traces in cells 1C (top left), 8A (top right) and 1A (bottom)

Network density

Cell density in the scenarios generated by the simulator is also an important metric. With the creation of 50 different scenarios, the results obtained say that the mean coverage in the network is equal to 4.05 cells, being the maximum coverage equal to 4.65 and the minimum 3.62, which denotes a dense network.

With this data, we know that if a user is connected to a cell that is later turned off for energy saving reasons, on average, it will be able to connect to other 3 cells. This way, we are minimizing the ratio of unserved users. Provided that there is an intelligent algorithm that switches cells off depending on the state of their neighbours, we can guarantee a certain level of connectivity for the users.

Cell switching: application of the Machine Learning models

The integration of the ML part of the project in the simulator is done when implementing the functionality of dynamically switching off cells depending on the state predicted by the classifier for the following 15-minute period. To do so, first a sample of one week in a scenario with 20 synthetic cells was generated. With those traces, the classification algorithms were trained in order to obtain an appropriate Random Tree model that would later be used in a large number of simulations to classify cells into the three aforementioned cell states.

The optimum model is a simple tree that only considers one sample in the past, as in the graph depicted in Fig. 4. 12. As a first approach, we will switch off all cells classified as OFF. We will be referring to this model as ***naïve***, as it is an aggressive algorithm that only considers the state of each independent cell.

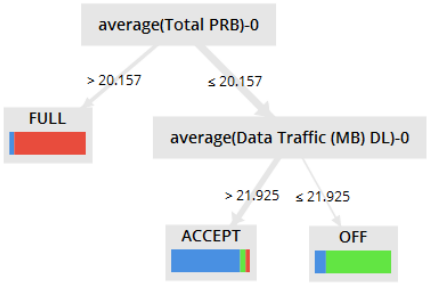


Fig. 4. 12 Example of Random Tree extracted from the traces

On the other hand, in order to add a further level of intelligence to the system, the naïve algorithm was modified to obtain the ***neighbour-aware*** one. This second approach differs from the previous one in the sense that, when deciding whether to switch a cell off or not, it takes into account the state of the neighbouring cells (in a radius of 500m). Only if more than a given percentage (which can be tuned to modify the restrictions of the system) of the neighbouring cells is still on, the studied cell can be powered off. That way, we increase the probability that a user associated to a cell that is turned off can connect to another neighbouring cell.

Without the application of the mentioned algorithm, an example result of the execution of the simulator for one day is represented in Fig. 4. 13. The graph has 96 samples, corresponding to 24 hours with a sample period of 15 minutes, and starting at 00am; on the vertical axis, the 90 cells (30 trisectorial eNB) are represented as a cumulative graph, being the ON (*ACCEPT* and *FULL* states) cells represented in blue and the OFF ones in orange. As depicted in the figure, the amount of cells being turned off follows a similar pattern to that seen in the cell profiles: the cells that must be on are about 70% during day hours, while at night, between 1 and 8 am approximately, the PRB utilization in most of the cells falls below the threshold that classifies them as unnecessary.

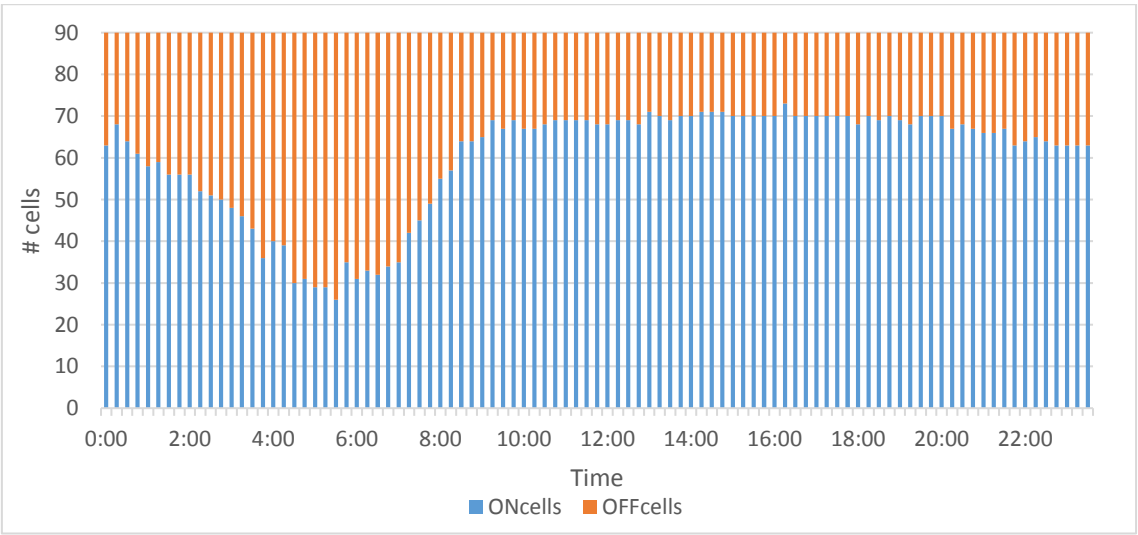


Fig. 4. 13 Cell activity in a simulation of one day, with the naïve algorithm

Going a step further, if we activate the neighbour-aware algorithm described some paragraphs above, with the restriction that, in order to switch off a cell classified as OFF, more than 70% of the surrounding cells must be ON, the results change in an interesting way. Several values were tested, and this figure (i.e. 70%) ended up being a good trade-off between power saving and user service, because higher values were too restrictive and therefore the power saving was minimal, while lower values resulted in behaviours similar to the naïve algorithm (all this additional graphs can be consulted in Appendix C.2). Fig. 4. 14 represents the same scenario as in the previous graph but, in this case, we are introducing the mentioned algorithm. This way, we have a more robust scenario, where the amount of cells being turned off is reduced, resulting in a smaller amount of users being unserved, as will be explained later. On the other hand, less power saving is achieved, too.

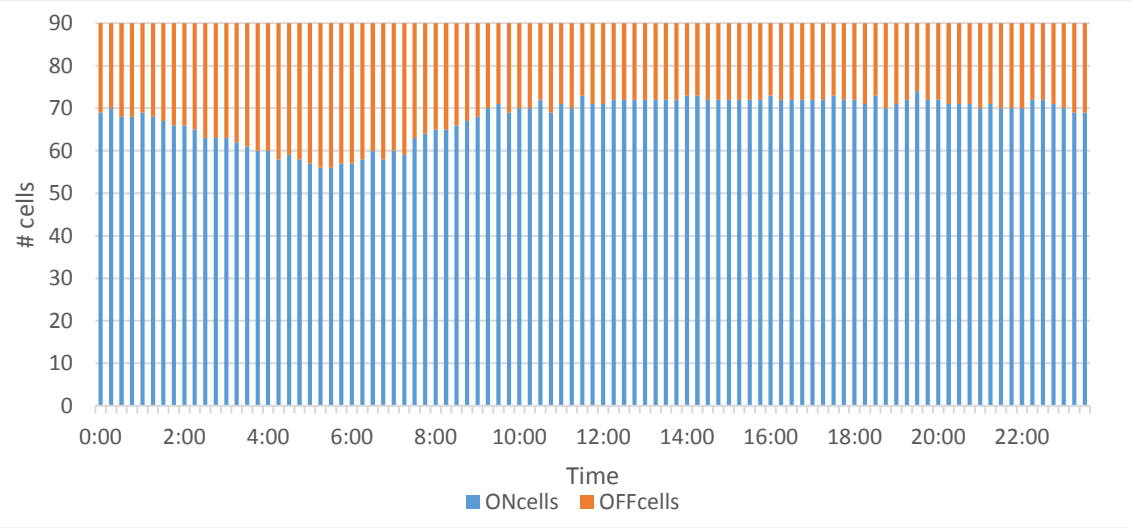


Fig. 4. 14 Cell activity in a simulation of one day, with the neighbour-aware algorithm

UE migration

When switching off a cell, the UEs that were connected to that cell must be migrated to another one serving that area. The algorithm that migrates them, tries to connect them to the cell with which they will have a better connection (i.e. the best MCS, as a ratio between distance to the eNB and its coverage radius) that is in *ACCEPT* state. If no neighbouring cell is in that state, cells in *FULL* state are considered; and finally, if there is no cell available, the user is left unserved, losing connectivity to the network (this problem can also happen in the limits of the scenario, where no other cell is available [see the limits of Fig. 3. 10], which would not happen in the real scenario because there would be continuous coverage).

Recovering the examples in the previous figures, the graphs below represent the total amount of migrated (green) and unserved (red) UEs that result from switching off cells. In the first case, Fig. 4. 15 shows how a naïve approximation working only with the decision tree results in an average of 3.2 UEs being migrated every 15 minutes (a 3.1% of the total UEs in the simulation), and 0.44 ending up unserved due to a total loss of coverage. These phenomena is concentrated during the hours with less activity (i.e. nightly hours) since that is

the period with a larger number of cells switched off. On the other hand, a more intelligent system as the one used in Fig. 4. 16 reduces these figures to an average of 2.5% of the total UEs migrated and only 0.3 unserved UEs on average. With the neighbour-aware algorithm activated, nearly no user is unserved (and spread over the whole day), while the naïve system had many more not-served users, especially concentrated during the night.

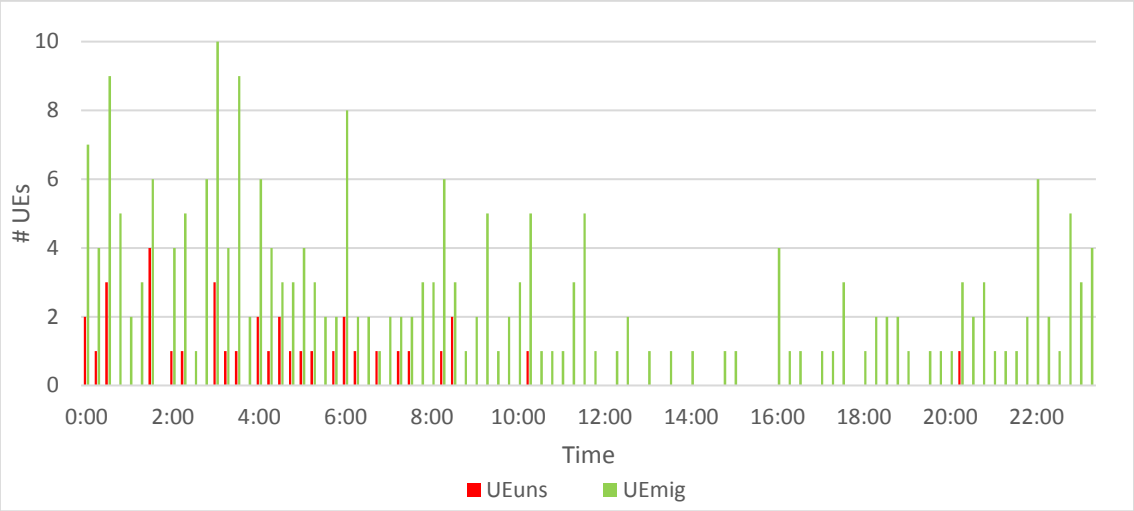


Fig. 4. 15 Total migrated and unserved UEs in a simulated scenario with naïve switching algorithm

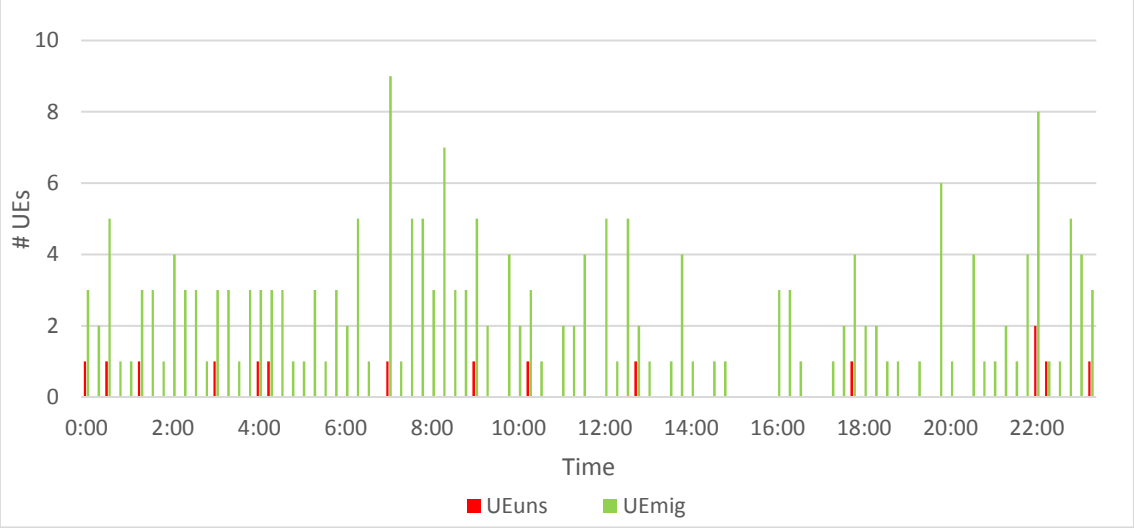


Fig. 4. 16 Total migrated and unserved UEs in a simulated scenario with the neighbour-aware algorithm

With the results shown in the this section, we can conclude that, by designing more sophisticated algorithms, we would be able to achieve really good results in terms of minimizing the amount of UEs that end up being unserved after an unsatisfactory migration. However, this can also affect the power saving metrics, as will be studied in the following section. Moreover, we must consider the case of a multi-tier (or layered) network where a macrocell covers the proposed small cells deployment, and therefore no UE will end up being unserved.

4.2.2 Power saving

As a final application of the whole thesis development, we planned on obtaining an intelligent system capable of reducing the power consumption in a mobile network thanks to the use of ML techniques alongside with the data being collected in that same network. The results detailed in the previous sections introduce the basis to understand the power saving metrics achieved with the proposed techniques.

In order to measure the power efficiency achieved with the results in this project, we have used the values proposed in [19], summarized in Table 4. 1, being δ equal to 0.1, as proposed in that same paper too.

Table 4. 1 Parameters for eNodeB power consumption model ([19])

eNodeB type	$P_{eNB,max}$ [W]	$P_{in,A}$ [W]	$P_{in,I}$ [W]
Macro	40	336.3	238.4 · δ
Micro	1	152.4	129.3 · δ
Femto	0.1	16.6	14.4 · δ

Cells that are either in *ACCEPT* or *FULL* state are categorized as active, and therefore have a power consumption following the values in column $P_{in,A}$; those cells that are in *OFF* state are not really turned off, but in an idle state, with some components off and some others on and, therefore, the power consumption can be really low, following the column $P_{in,I}$.

With this metrics, the results of the total power consumption in the network are the values represented in Fig. 4. 17. The orange line represents the maximum power consumption, assuming a scenario where macro, micro and femtocells are equally distributed among the 90 total cells (90 cells by 168.43W on average). The blue curve shows the evolution of power consumption over time in the simulated scenario with the application of the naïve algorithm. The graphs show how, on average, 5,609.8W are being saved (sampling every 15 minutes), which implies that during a whole day, the total power saving is approximately a 40% of the total power consumed during a day. The maximum power saving is, as expected, during nightly hours (when more cells can be switched off), with a peak of 10,210.7W, approximately twice the average savings. With this figures, we can estimate an average saving per day approximate to 134.4kWh (5.6kW * 24h), which, according to the most recent values of the cost of energy of Endesa[6] (one of the most important companies in the power industry), corresponds to 18.97€ saved per day, making 6,923€ on a year.

[6] https://www.endesaclientes.com/articulos/tarifas-reguladas-luz-gas.html

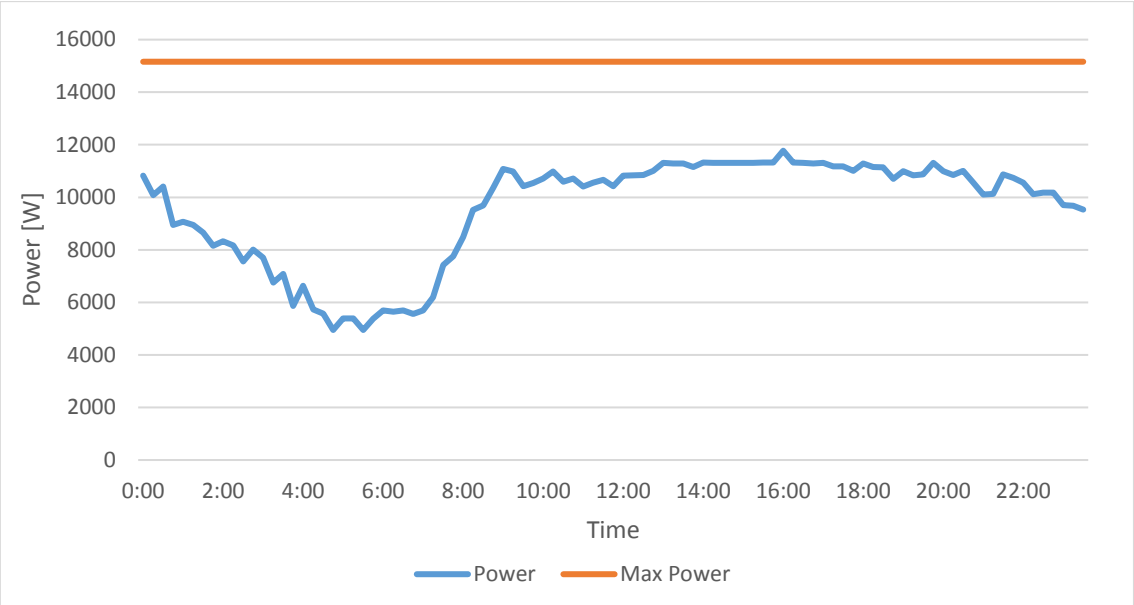


Fig. 4. 17 Power consumption during a day in a simulated scenario with a naïve algorithm

On the other hand, the incorporation of a more intelligent algorithm as the neighbour-aware aforementioned, ensures a more efficient management of the users, who will suffer less service losses from the network, as stated in the previous subsection. Taking the same scenario studied in previous sections, now the figures illustrated in Fig. 4. 18 show that the total power saving sums up nearly a 30% of the energy consumed daily, given that there is an average of 4,146W every 15 minutes (a nightly maximum of 6,077W, 1.5 times the average), approximately 99.5kWh a day or, in other words, 14€ a day and 5110€ a year.

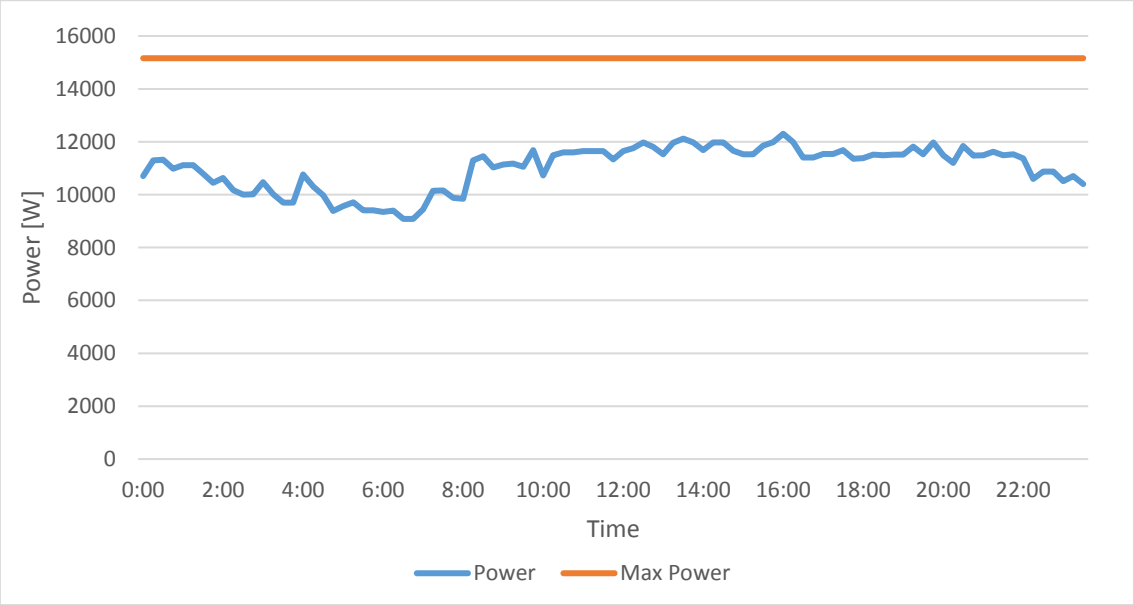


Fig. 4. 18 Power consumption during a day in a simulated scenario with a more sophisticated algorithm

A final comparison between the naïve algorithm and the neighbour-aware one with different thresholds can be found in Fig. 4. 19. As proved in earlier

paragraphs, the naïve algorithm provides good results in terms of power saving (around 25% on average) but does not perform so well in terms of unserved UEs due to many cells being switched off. On the other hand, the neighbour-aware algorithm can have different configurations that make it either more aggressive, with decent power savings, or more conservative, with few disconnected UEs.

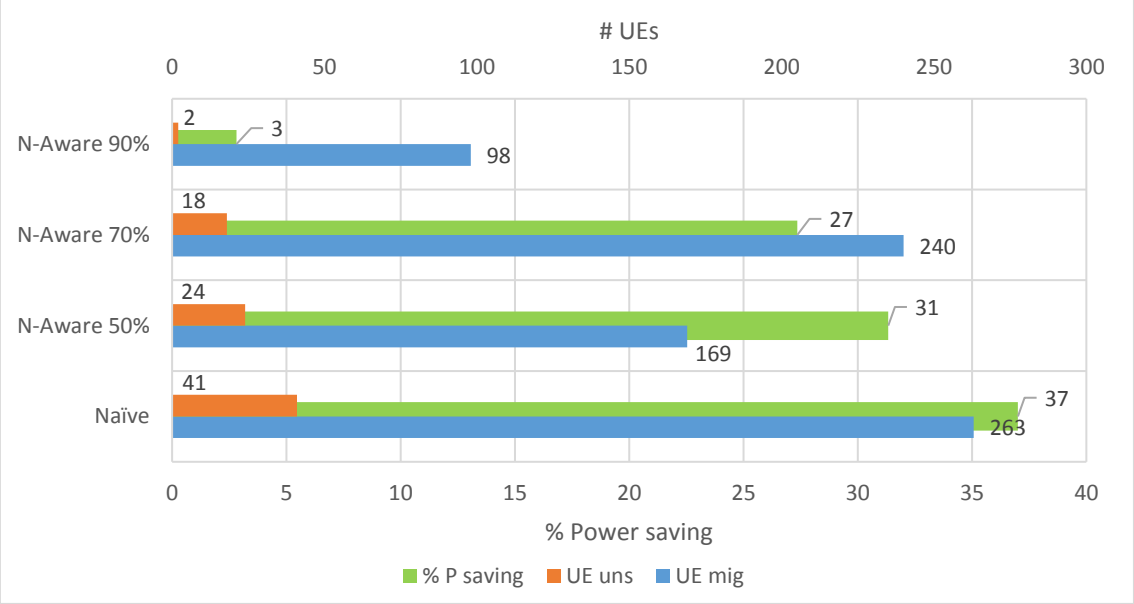


Fig. 4. 19 Comparison of migrated UEs and power saved with different switching strategies

Table 4. 2 summarizes some of the most relevant results obtained with the simulation scenarios proposed in this thesis. The figures have been rounded and compared to further simulations in similar scenarios. The simplest scenario is the one in which no dynamic switching algorithm is applied, where, obviously, all the users are being served (in an ideal scenario) and the power consumption is maximum. In a first approximation to a naïve dynamic algorithm, good results are obtained, as high power saving figures are achieved (over one third of the total consumption) in expense of pushing some users out of service during night hours due to the high amount of cells being turned off. Finally, a more sophisticated algorithm is proposed, where cells consider their neighbours before taking the decision of switching off, and a better ratio of unserved users is obtained, although then fewer cells are off and therefore the power saving is lower.

Table 4. 2 Summary of results from the simulations

	Naïve	Neighbour-aware	No algorithm
Cell density	4 cells/point	4 cells/point	4 cells/point
# Off cells	High	Medium	None
Unserved users	< 0.6%	< 0.4%	0%
Power saving	≈ 40%	≈ 25%	-
Energy saving	130 kWh/day	90 kWh/day	-
Expense cut	6000 €/year	4500 €/year	-

CHAPTER 5. CONCLUSIONS AND FUTURE WORK

Regarding the development of this master thesis, the overall result is positive, because the main objectives of the project have been accomplished, while most of the challenges that have been appearing during the development could be overcome.

First of all, there was a thorough study of how to apply Machine Learning techniques in network management, consulting many sources in order to learn as much as possible and obtain interesting and useful knowledge. Later on, and with the help of the research team in i2CAT, there was some work on data processing and application of ML techniques to the data sets of a real network provided by COSMOTE. During several weeks, progress was made in terms of defining the input sets to solve our initial problem and finding the best possible prediction algorithms to different parameters that could be interesting for later stages of the project.

On the second part of the thesis development, we thought it would be interesting to test the results in a simulated scenario where different tests could be made, in order to validate the performance of the previously obtained models, and work on possible applications of such proposals. With a wide variety of options in front of us, the proof of concept chosen for this contribution to the 5G-XHaul project consisted on evidencing the energy savings that could be obtained thanks to network management strategies based on ML modelling, without affecting the performance experienced by the users. With that objective in mind, a Java-based simulator was programmed, capable of generating random scenarios based on the real traces that collect different metrics related to cell switching and user migration.

Finally, once everything was ready, and the two parts of the project were integrated in a single environment, conclusions and results could be obtained, thus resulting in the proof that taking profit of the data collected in a cellular network can have many applications, one of them being a dynamic switch-off system that can turn cells off during valley hours with the aim of saving power, while maintaining connectivity for the users of the network. We must also consider the case of this kind of strategy being applied over a multi-tier or layered network, where there is a macro cell with a large coverage area over the small cell deployment proposed in this thesis. In such case, no UE would suffer a complete disconnection, as they would already have connectivity to the network, so unserved users is not such a big problem.

In a more personal note, I would like to say that the development of this project has been very gratifying. Taking part again in a bigger project where I can feel that I can provide useful things, while putting into practice some of the knowledge acquired during my studies and learning many other new things, is really a good experience. This project has helped me get deeper in the field of Data Analysis and Machine Learning, and understand how interesting it can be, and how complex it can get when you are looking for specific results in a certain area. Moreover, with a project like the one presented in this thesis, you can notice the

importance of this new field related to data processing, and how it is not only interesting by itself, but mainly due to the many areas it can be applied to and how each of them can have completely opposite requirements that can, however, be extracted with the same techniques.

From a wide and generic perspective, having completed a project like this one, where the initial expectations were pretty high and the results are in accordance with those initial requirements, I can say I am really happy with the results.

5.1. Future work

Despite having fulfilled all the main objectives that were set up at the beginning of the project, there is still work to do in this field, and not only related to the bigger 5G-XHaul project to which this thesis has contributed. The team is focused in exploiting the data collected from the network to provide intelligent network management strategies that can result in different benefits; in this thesis we have only tested the application of such techniques to energy saving algorithms, but there are many other interesting alternatives that can take profit of similar data processing mechanisms.

Regarding the state of the art of ML techniques in network management, during the early stages of the project we learned that there are lots of research teams that have been working in the field, with really interesting results which not always have been put into practice, so it is a really interesting work to do to bring that knowledge to real scenarios where the benefits can be made real.

In terms of the final results of the project, we have found them really interesting and promising, but there is still a long way to go. Here we have worked with simplified algorithms that provide good results, but many different techniques and strategies could be applied in order to achieve even better results. Just as an example, this same cell switching algorithm could take into account the load on neighbouring cells to perform load balancing in the network and relieve heavily loaded eNodeBs. There are as much applications as one can think of, and it is a matter of time that someone will start exploring them.

On the other hand, there have also been not so satisfactory key points during the development, as there have been some limitations in the project that make the results interesting but not as good as could be. The final results show a yearly economic and energetic saving in the network that may not pay off the effort put in the algorithmic. Maybe the combination of the proposed strategies with alternative network deployments oriented towards energy efficiency could improve those figures significantly. Moreover, the resources with which we have had to work may have limited the results too. For example, regarding the developed simulator, despite being based on the real traces, there are some features that could be improved (such as the UE generation) if more detailed information had been available. So, as a general view, many challenges have also been confronted during the development of this thesis, and there are still interesting tasks that can extend the project contributions.

With regard to the environmental impact of the project that has been developed, there is not much to say, as the current legislation makes sure that cellular networks work within non-dangerous transmission power limits, plus small cell scenarios such as the ones that will be deployed when 5G networks go into the streets tend to have even lower power levels. Moreover, one of the general ideas of the project itself is providing an algorithm that dynamically turns cell off, which reduces transmissions and power consumption, in line with the energy-efficient trend that is coming with future generation networks.

Finally, in terms of ethical considerations of the project, we may highlight that they are positive, as the proof of concept presented in this document shows how energy savings can be obtained by the design of intelligent systems that dynamically adapt to the requirements of the network at any given moment. It also has an impact in economic terms, due to the savings associated to the reduction in power consumption, but there is no influence further than the management of the network itself and the reduction in energy consumption.

All in all, this thesis covers many relevant aspects about the application of Machine Learning techniques to network management, and presents possible strategies which, making use of the data that is already being collected in cellular networks, can be part of the future networks. Finally, some experiments in a custom simulation environment were run in order to prove the benefits of such strategies.

BIBLIOGRAPHY

[1] 5G-XHaul project website - http://www.5g-xhaul-project.eu/index.html

[2] Horizon 2020 project website - http://ec.europa.eu/research/fp7/index_en.cfm

[3] I2CAT Foundation website - http://www.i2cat.net/en

[4] Mitchell, T. (1997), “Machine Learning”, McGraw Hill. p. 2.

[5] Agrawal D. et al., “Challenges and Opportunities with Big Data: A white paper prepared for the Computing Community Consortium committee of the Computing Research Association”, 2012

[6] Big Data definition by Oracle - https://www.oracle.com/uk/big-data/index.html

[7] Oracle, “An Enterprise Architect’s Guide to Big Data”, White Paper, March 2016

[8] Amazon Web Services - https://aws.amazon.com/es/ec2/

[9] Micorsoft Azure - https://azure.microsoft.com/es-es/

[10] OpenStack, an open-source cloud service - https://www.openstack.org/

[11] Hebbian theory for Neural Networks - https://en.wikipedia.org/wiki/Hebbian_theory

[12] Perceptron - https://en.wikipedia.org/wiki/Perceptron

[13] AdaLine - https://en.wikipedia.org/wiki/ADALINE

[14] Pérez-Romero J., Sánchez-González J., Sallent O., Agustí R., “On Learning and Exploiting Time Domain Traffic Patterns in Cellular Radio Access Networks”, 2016

[15] Fjelberg, M., “Predicting data traffic in cellular data networks”, Master Thesis in KTH, June 2015

[16] Y. Zang, F. Ni, Z. Feng, S. Cui, Z. Ding, “Wavelet Transform Processing for Cellular Traffic Prediction in Machine Learning Networks”, ChinaSIP 2015

[17] H, Maciejewski, M, Sztukowski, B. Chowanski, “Traffic Profiling in Mobile Networks Using Machine Learning Techniques”

[18] U. Paul, L. Ortiz, S. R. Das, G. Fusco, M. Madhav Buddhikot, “Learning Probabilistic Models of Cellular Network Traffic with Applications to Resource Management”, DYSPAN 2014

[19] K. Hiltunen, “Utilizing eNodeB Sleep Mode to Improve the Energy-Efficiency of Dense LTE Networks”, in Proc. of 24th IEEE Personal Indoor and Mobile Radio Communications (PIMRC), 2013

[20] P. Frenger, P. Moberg, J. Malmodin, Y. Jading, I. Godor, ”Reducing Energy Consumption in LTE with Cell DTX”, in Proc. IEEE Vehicular Technology Conference (VTC) 2011-Spring, Budapest, Hungary, 2011

[21] Wang et al., "SDN-based joint backhaul and access design for efficient network layer operations," 2015 European Conference on Networks and Communications (EuCNC), Paris, 2015, pp. 214-218

[22] RapidMiner official website - https://rapidminer.com/

[23] Apache Hadoop official website - http://hadoop.apache.org/

[24] RapidMiner Series Extension - https://marketplace.rapidminer.com/UpdateServer/faces/product_details.xhtml?productId=rmx_series

ABBREVIATIONS AND ACRONYMS

3GPP	3rd Generation Partnership Project
AI	Artificial Intelligence
API	Application Programming Interface
BBU	BaseBand Unit
BTS	Base Transceiver Station
CCC	Computing Community Consortium
C-RAN	Cloud Radio Access Network
CSV	Comma Separated Values
DB	Data Base
DL	Downlink
DS	Data Set
DTX	Discontinuous Transmission
EDGE	Enhanced Data Rates for GSM Evolution
eNB	Evolved NodeB
EPC	Evolved Packet Core
E-UTRAN	Evolved - UMTS Terrestrial Radio Access Network
GMM	Gaussian Mixture Model
GSM	Global System for Mobile Communications
HSPA	High Speed Packet Access
ITU	International Telecommunication Union
Lasso	Least Absolute Shrinkage and Selection Operator
LTE	Long Term Evolution
LTE-A	Long Term Evolution – Advanced
MCS	Modulation and Coding Scheme
ML	Machine Learning
MME	Mobility Management Entity
NFV	Network Function Virtualization

OLS	Ordinary Least Squares
P-GW	PDN Gateway
PRB	Physical Resource Block
PUCCH	Physical Uplink Control Channel
PUSCH	Physical Uplink Shared Channel
QoE	Quality of Experience
RE	Relative Error
RF	Radio Frequency
RMSE	Root Mean Square Error
RRH	Remote Radio Head
SDN	Software Defined Network
SDR	Software Defined Radio
S-GW	Serving Gateway
SNIR	Signal to Noise and Interference Ratio
SVM	Support Vector Machine
THR	Throughput
TTI	Transmission Time Interval
UE	User Equipment
UL	Uplink
UMTS	Universal Mobile Telecommunications System

ANNEXES

APPENDIX A. RapidMiner results

A.1. Regression prediction

Comparison of real and predicted traces for throughput and number of users, with a simple regression algorithm. Further explanation of these graphs can be found in section 4.1.2.

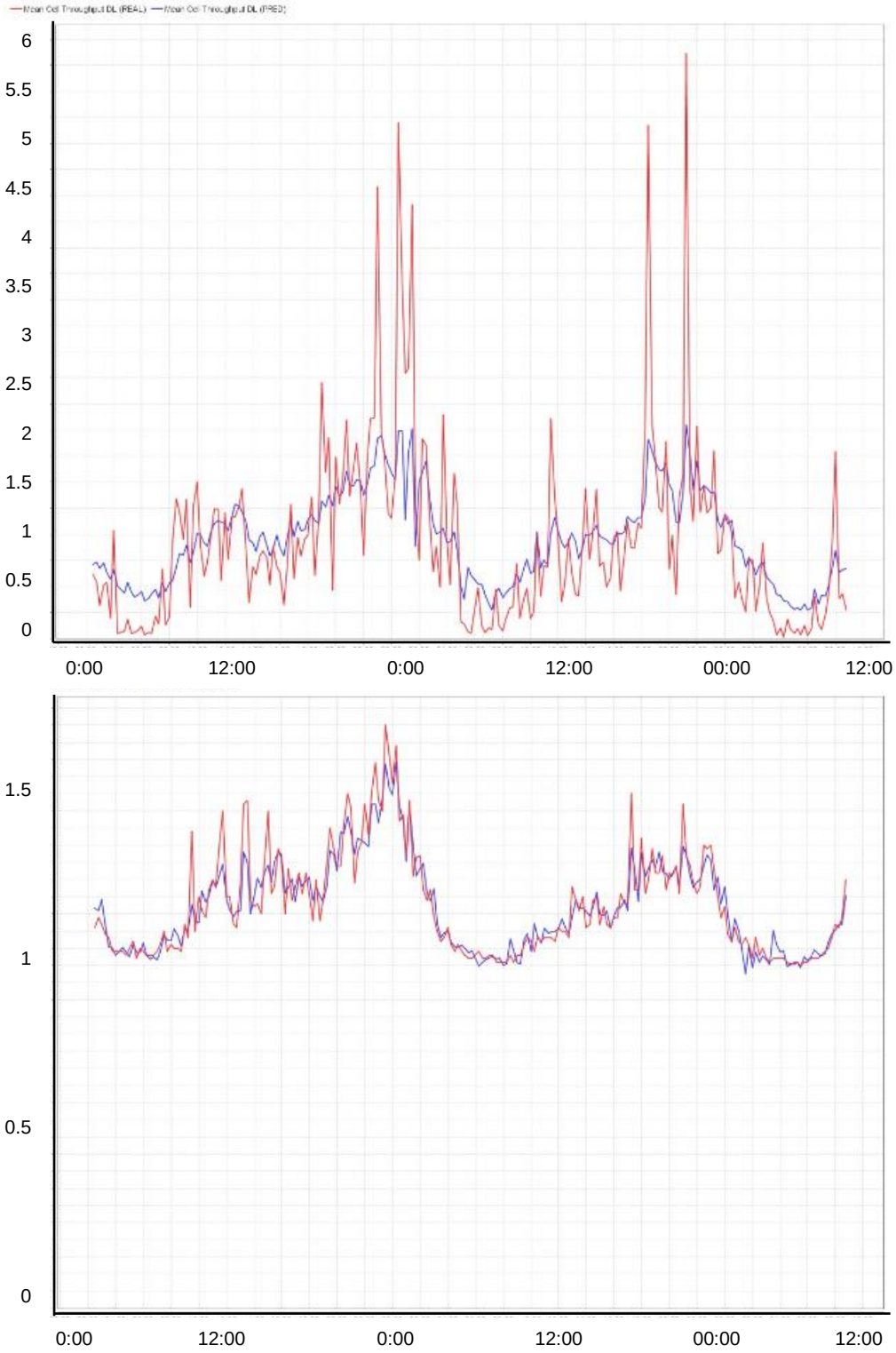


Fig. A. 1 Regression prediction of DL Throughput (top) and #UEs (bottom) of 3 days in cell 1C (real in red, predicted in blue)

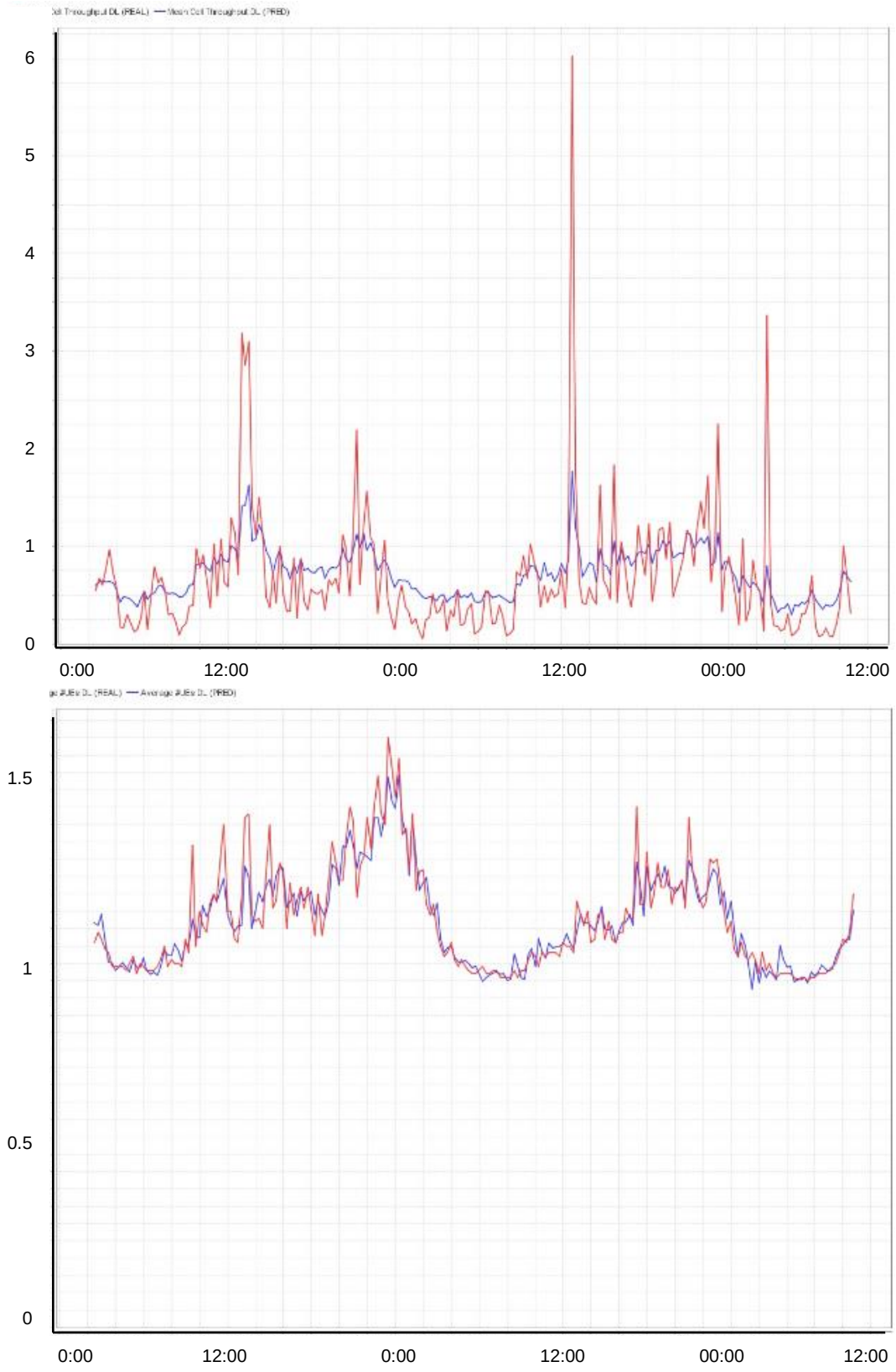


Fig. A. 2 Regression prediction of DL Throughput (top) and #UEs (bottom) of 3 days in cell 8A

A.2. Traffic aggregation

Comparison of real and predicted traces for throughput and number of users, with a more sophisticated technique, by aggregating (average) traffic to predict either aggregated traffic (Fig. A. 3) or the one from a single cell (Fig. A. 4). Further explanation of these graphs can also be found in section 4.1.2.

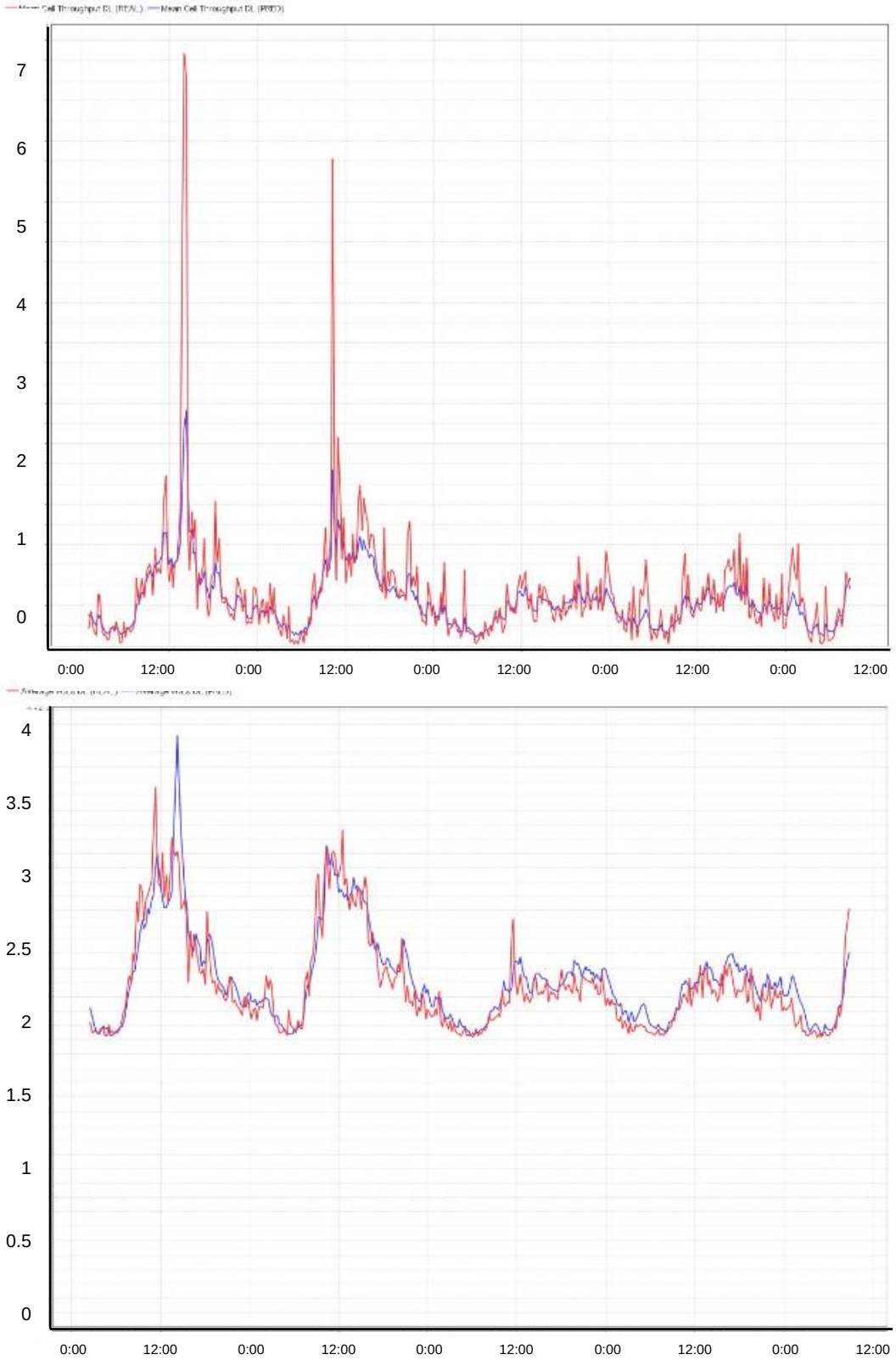


Fig. A. 3 Regression prediction of Throughput (top) and #UEs (bottom) of 3 days in eNodeB 3

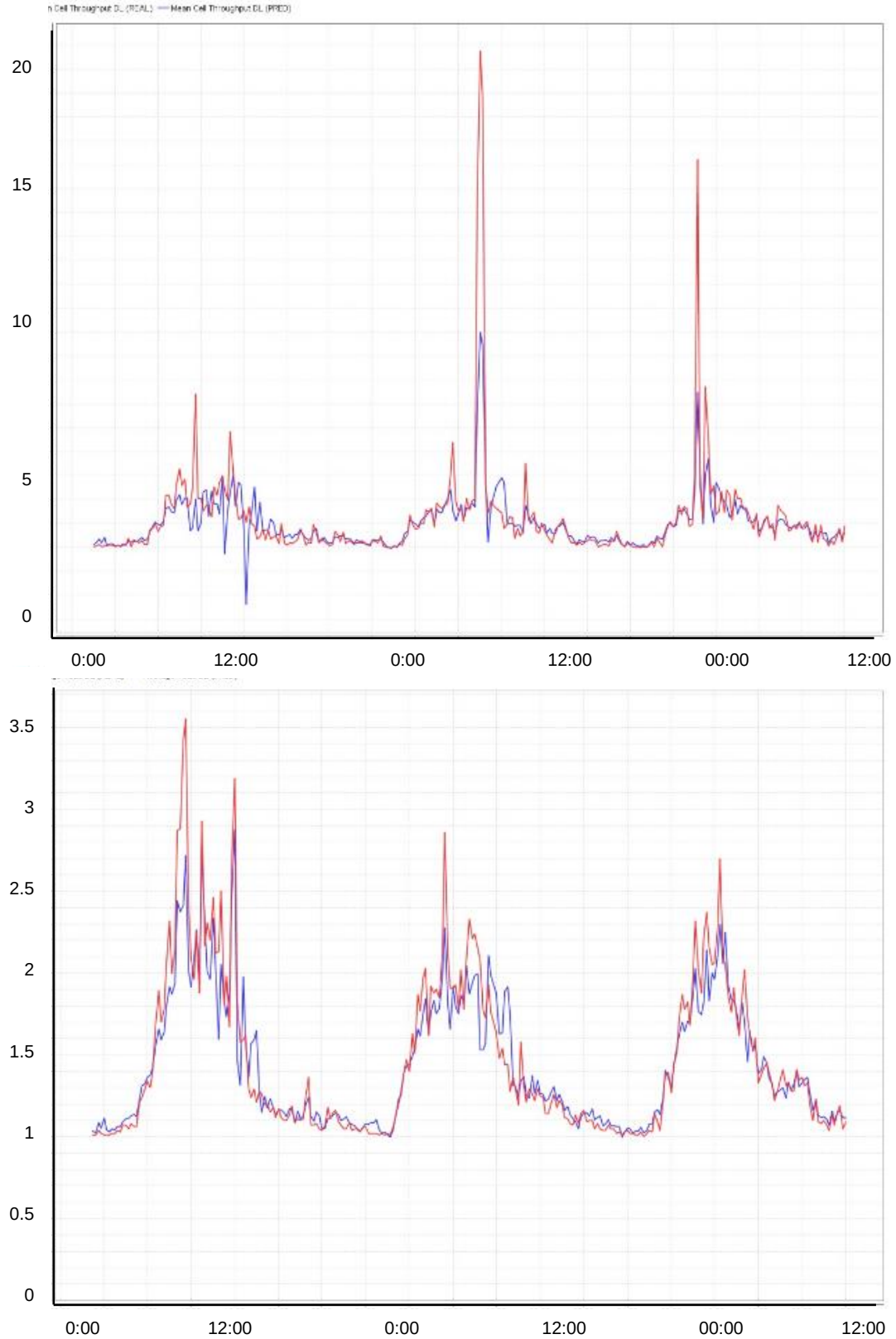


Fig. A. 4 Regression prediction of Throughput (top) and #UEs (bottom) using the aggregated traffic from two eNBs (1 and 3)

A.3. Separate weekdays and weekends

Comparison of real and predicted traces for throughput and number of users, when two different data sets are considered, one only including weekdays, and the only with weekends. Refer to section 4.1.2 to learn more about these graphs.

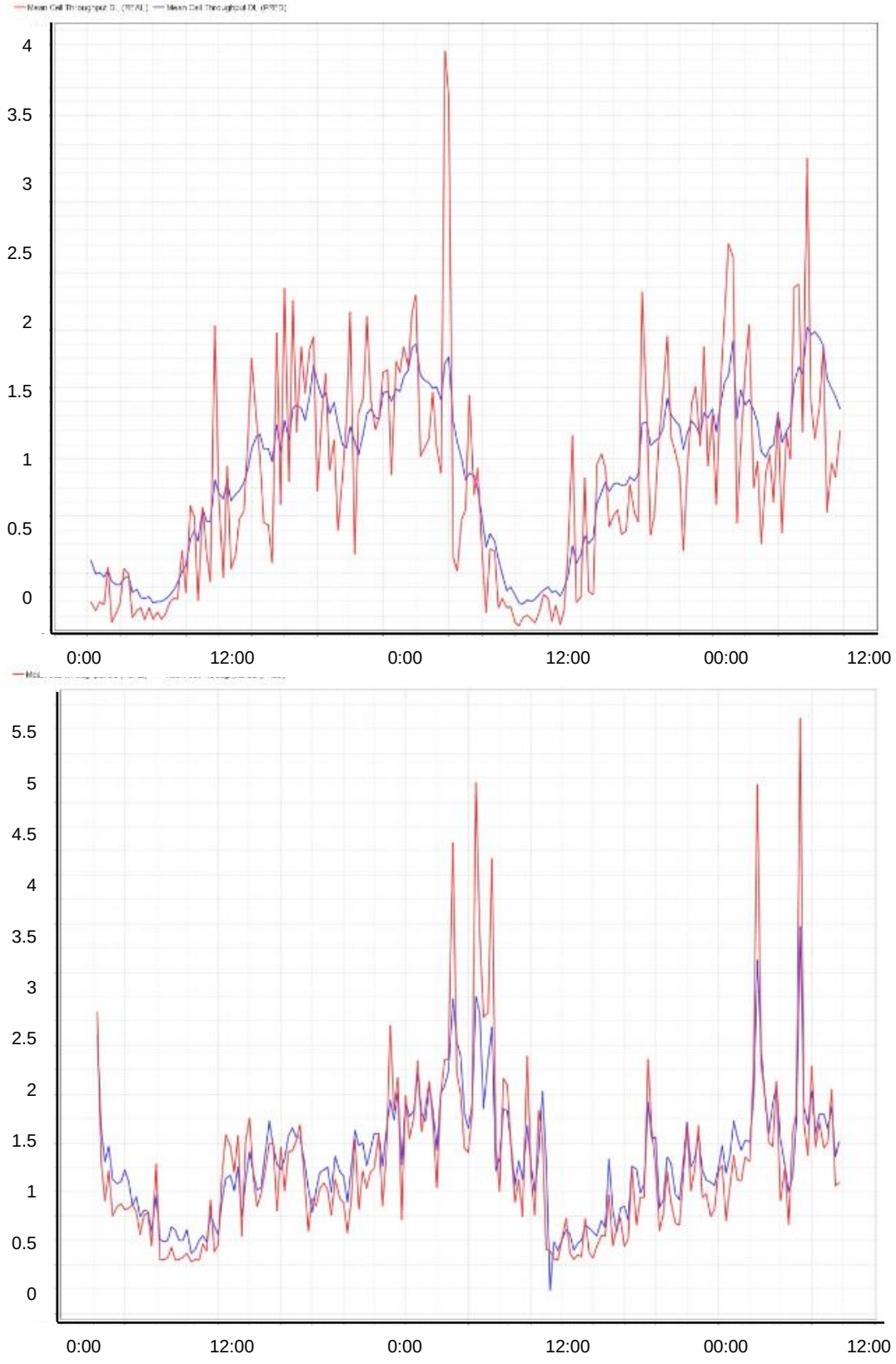


Fig. A. 5 Regression prediction of Throughput separating weekdays (left) from weekends (right)

A.4. PRB Prediction

Comparison of real and predicted traces for PRB utilization, with a simple regression algorithm. More information about the figures can be found in section 4.1.2.

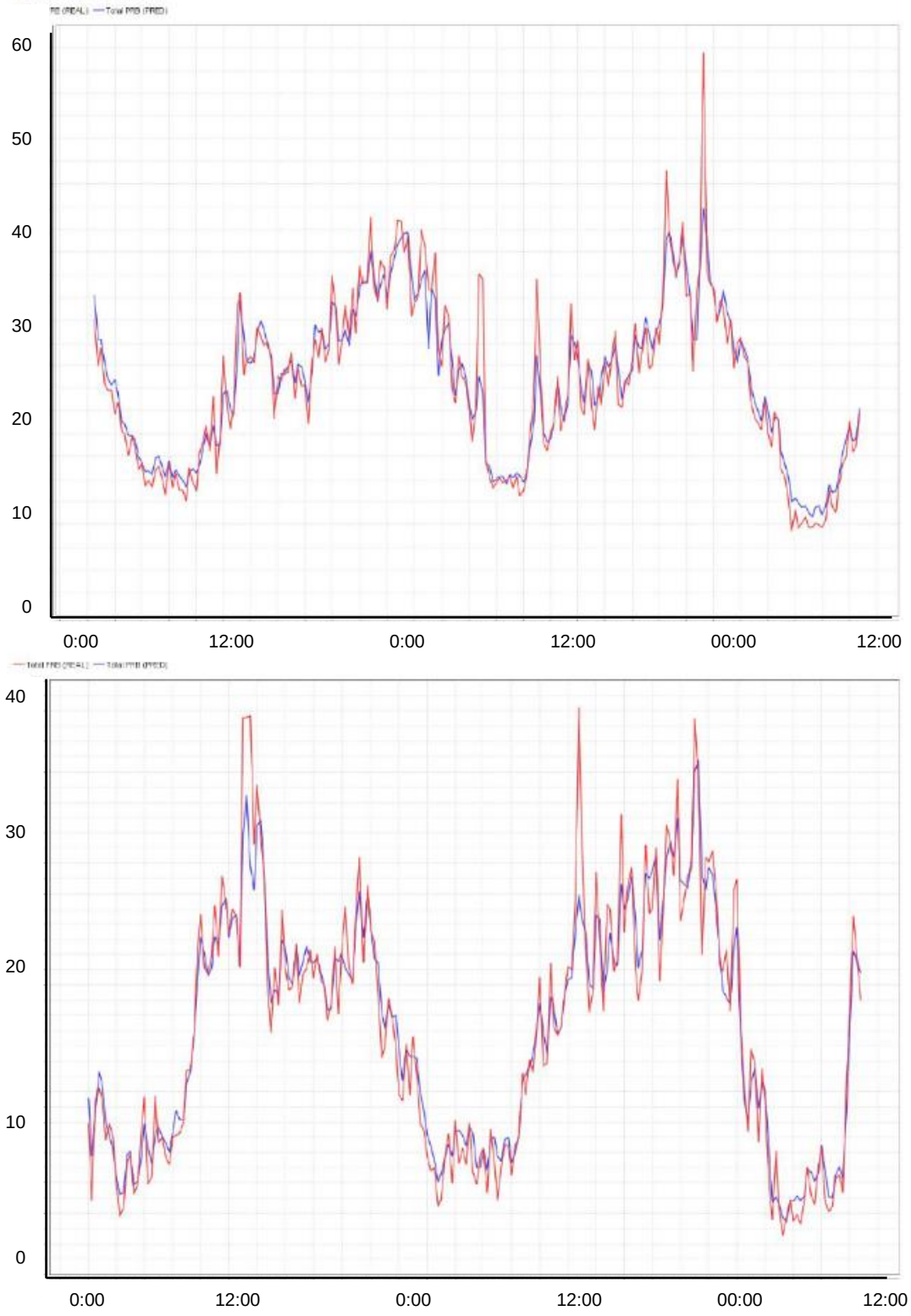


Fig. A. 6 Regression prediction of PRBs in cell 1C (left) and 8A (right)

A.5. Granularity reduction

Comparison of traces for PRB utilization when reducing granularity from 15-minute samples to 30-minute ones. A more detailed explanation of the graphs can be found in section 4.1.2.

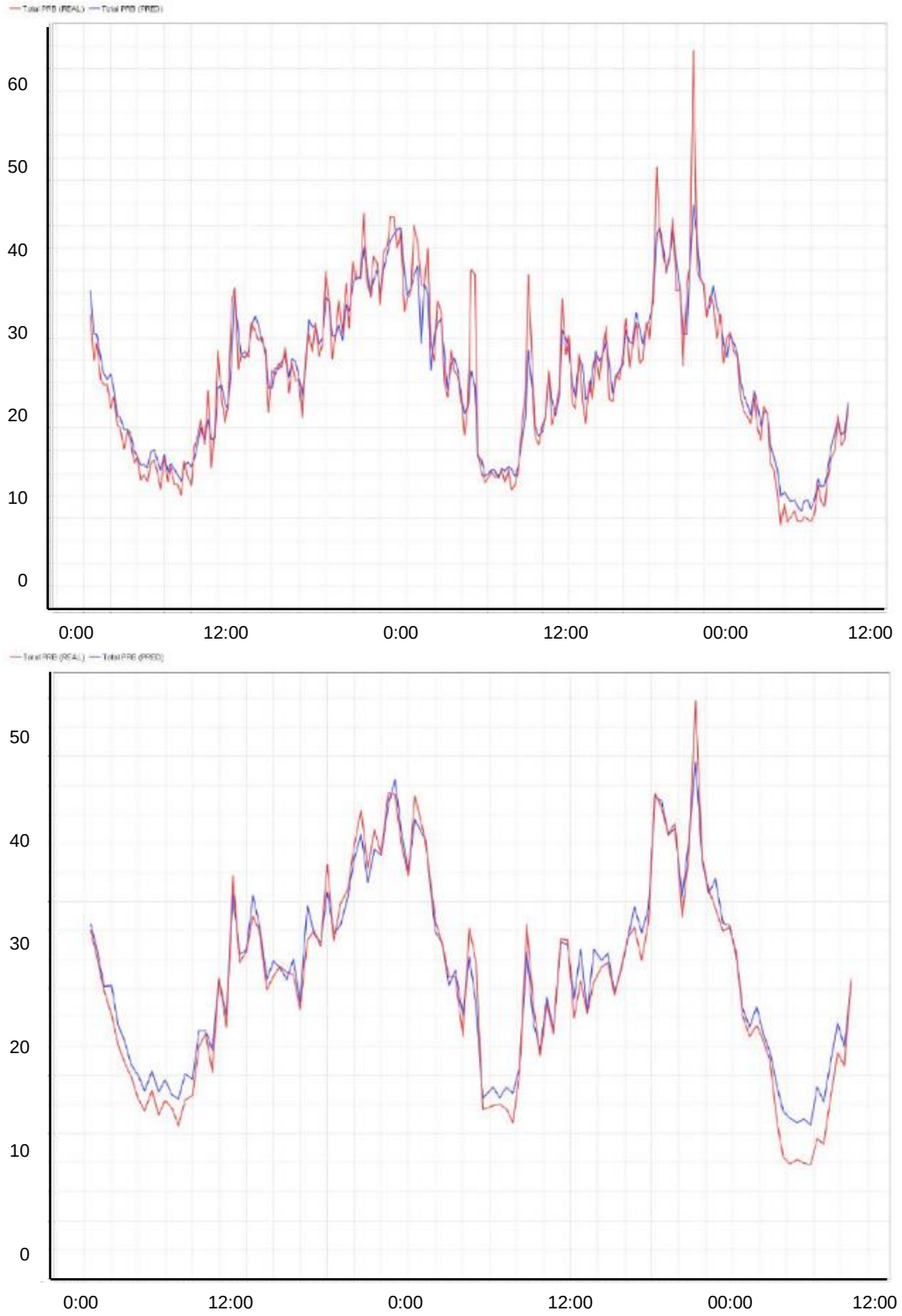


Fig. A. 7 Regression prediction of PRBs with granularity of 15 (top) and 30 minutes (bottom) (real values in red, predicted in blue)

APPENDIX B. Clustering results

B.1. Clustering

Detailed results of the clustering techniques applied over the cells in the data set. The table shows the cell ID, the cluster it belongs to (by colour), and the main features taken into account on the clustering. More information in section 4.1.4.

Cell	PRB average	PRB variance
1 (1A)	6.312	23.119
2 (1B)	8.570	40.686
3 (1C)	20.188	71.074
4 (2A)	10.133	17.821
5 (3A)	7.502	12.184
6 (3B)	3.020	5.262
7 (3C)	4.486	3.389
8 (4A)	13.310	11.740
9 (4B)	10.302	32.906
10 (4C)	3.283	5.544
11 (5A)	9.836	21.408
12 (5B)	6.314	8.736
13 (5C)	4.537	7.256
14 (6A)	14.210	51.745
15 (6B)	5.025	2.740
16 (6C)	0.619	0.245
17 (7A)	0.764	0.121
18 (7B)	0.554	0.053
19 (7C)	9.665	21.205
20 (8A)	6.945	10.988
21 (8B)	5.102	24.610
22 (8C)	2.965	1.895
23 (9A)	2.086	2.989
24 (9B)	7.952	11.751
25 (10A)	2.556	2.729
26 (10B)	5.037	9.728
27 (10C)	0.458	0.052

Fig. B. 1 Cells clustered by load: very high (red), high (green) and medium (yellow)

APPENDIX C. Simulation results

C.1. Result of synthetic traces

Comparison of the result of real and synthetic traces obtained with the simulation environment. Further information can be found in section 4.2.

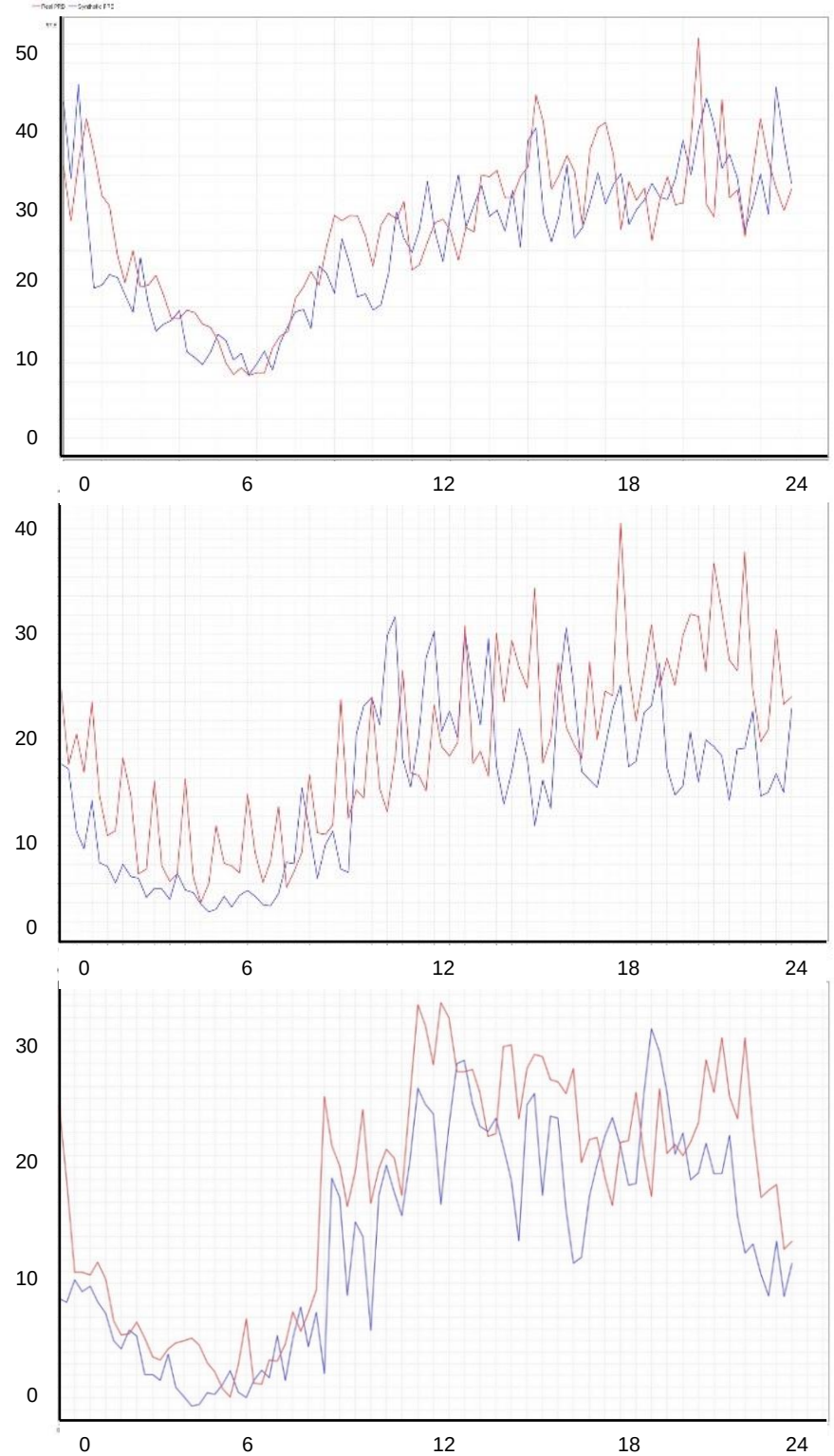


Fig. C. 1 Comparison of real (red) and synthetic (blue) traces in cells 1C (top left), 8A (top right) and 1A (bottom)

It is important to notice that, as the scenario has more cells than those of the data sets of the real network, more than one simulated cell may follow the traces of a common real cell. In such case (Fig. C. 2), the traces are also different between them, which ensures that no repeated pattern is found in the whole simulated scenario. The image shows two cells from a simulation (cells 40 and 58) that happen to be based on the same real cell, 3C. As depicted in the figure, their traces are somehow similar, but present some slight variations in the profile, as well as some other bigger changes.

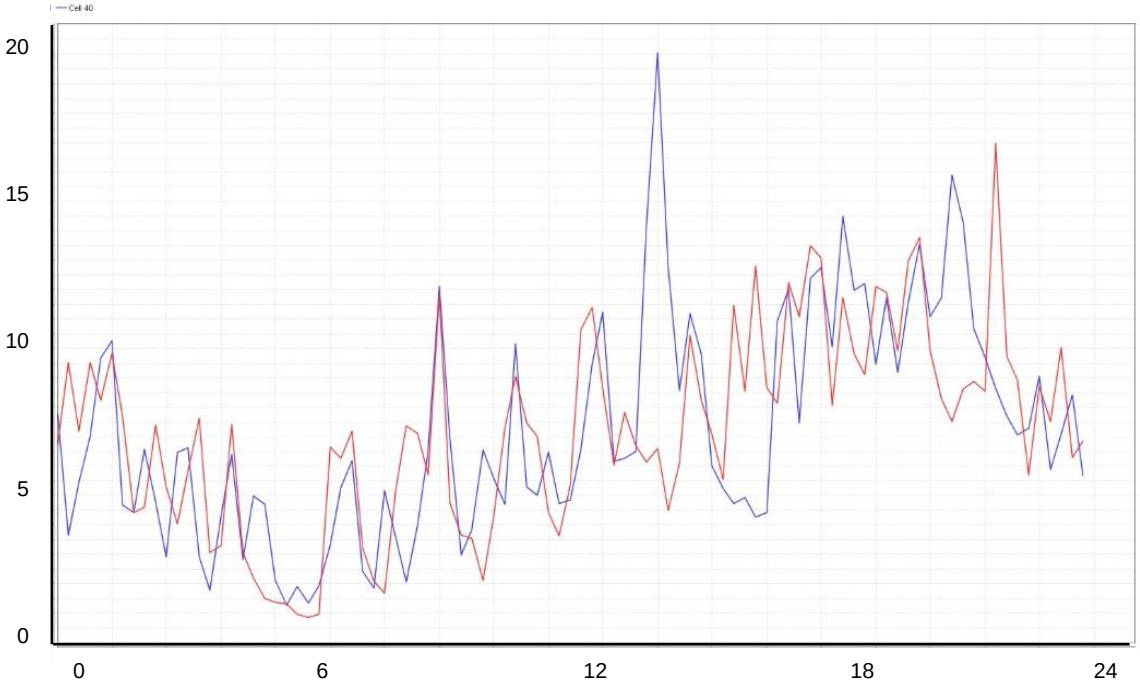
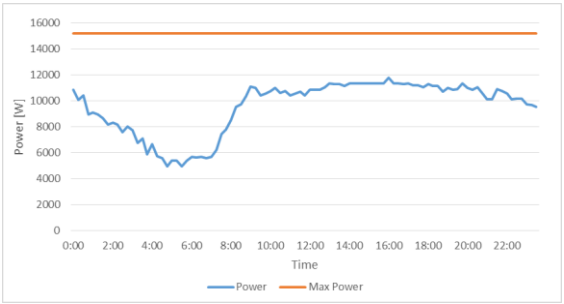
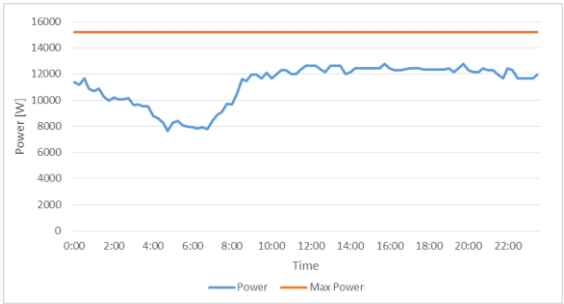


Fig. C. 2 Comparison of cells 40 (blue) and 58 (red) in a simulation, both based on the traces of the real cell 3C

C.2. Comparison of switching algorithms

Comparison of the results regarding migrated users and overall power consumption during a whole simulated day using the different cell switching algorithms that have been presented in this thesis. Further information regarding this topic is found in section 4.2.2.





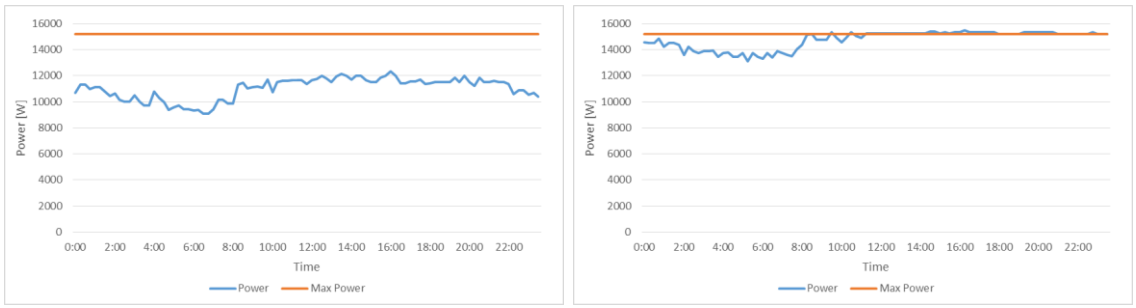


Fig. C. 3 Comparison of power consumption during a day in a simulation with a naïve algorithm (top left), and neighbour aware with a threshold of 50% (top right), 70% (bottom left) and 90% (bottom right)

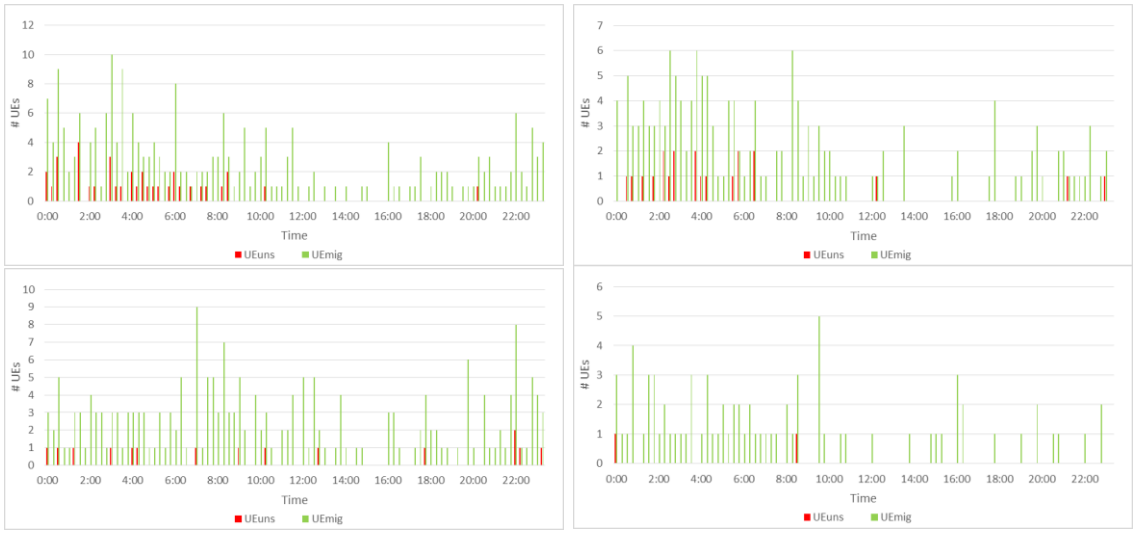


Fig. C. 4 Comparison of migrated (green) and unserved (red) UEs during a day in a simulation with a naïve algorithm (top left), and neighbour aware with a threshold of 50% (top right), 70% (bottom left) and 90% (bottom right)